(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号

特開平10－79042

(43)公開日　平成10年(1998)３月24日

| (51)Int.Cl.[6] | 識別記号 | 庁内整理番号 | ＦＩ | | 技術表示箇所 |
|---|---|---|---|---|---|
| Ｇ０６Ｔ　13/00 | | | Ｇ０６Ｆ　15/62 | ３４０Ａ | |
| Ｈ０４Ｎ　5/262 | | | Ｈ０４Ｎ　5/262 | | |

審査請求　有　　請求項の数17　ＯＬ　（全 38 頁）

(21)出願番号　特願平8－233492

(22)出願日　平成８年(1996)９月３日

(71)出願人　396001980
株式会社モノリス
東京都港区麻布十番１丁目７番３号

(72)発明者　伊藤　博文
東京都港区麻布十番１丁目７番３号　株式会社モノリス内

(72)発明者　品川　嘉久
東京都江戸川区西葛西５－10－26－204

(72)発明者　國井　利▲やす▼
東京都文京区本郷１－25－21　ドムス本郷602

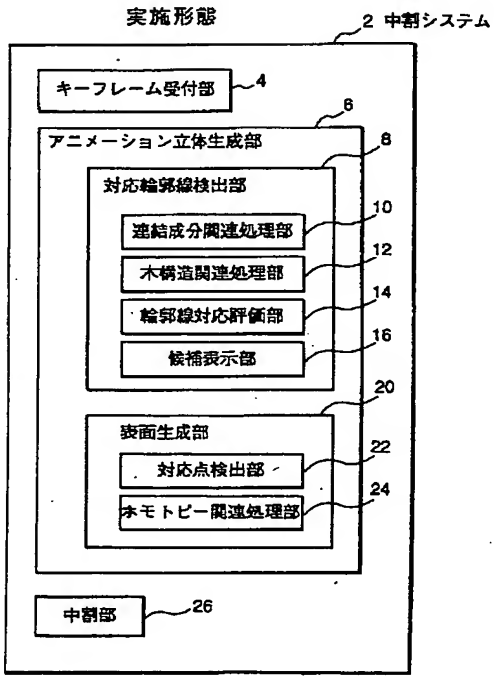
(74)代理人　弁理士　吉田　研二　（外２名）

(54)【発明の名称】　アニメーション処理方法およびその応用

(57)【要約】

【課題】　従来のアニメーションの中割システムは位相が変化するような場合に対応できず、用途が限られていた。また、操作性も悪かった。

【解決手段】　アニメーション立体生成部６をもつ。アニメーション立体とは、高さｚ方向に時間ｔをとり、平面ｔ＝ｔ０による断面がその時刻におけるフレーム画像になるような立体である。この立体の生成に当たり、連結成分や輪郭線の木構造など、位相を考慮する。この立体の断面を切り出せば中割が実現する。位相幾何学的な根拠はモース理論に求める。



1

【特許請求の範囲】

【請求項１】　フレームをキーフレームとそれ以外の中間フレームに分類して処理を行うアニメーション処理方法において、

オブジェクトが描かれた複数のキーフレームを準備するキーフレーム準備工程と、

アニメーション立体を生成する工程であって、その立体の断面の輪郭線形状を前記キーフレームに描かれたオブジェクトの輪郭線形状に一致させたうえで、その立体の位相に注目しながらその立体の表面を生成する立体生成工程と、

を含むことを特徴とするアニメーション処理方法。

【請求項２】　請求項１に記載の方法において、該方法はさらに、

生成されたアニメーション立体の前記断面とは異なる断面における輪郭線形状をもとに中間フレームに含まれるオブジェクトの二次元形状を取得する補間工程を含むことを特徴とするアニメーション処理方法。

【請求項３】　請求項１に記載の方法において、

前記立体生成工程は、

隣接するキーフレームに描かれているオブジェクトの輪郭線どうしの近さをもとに隣接するキーフレーム間における輪郭線の対応関係を把握する対応輪郭線検出工程と、

対応する輪郭線間の表面をホモトピーの概念を用いて生成する表面生成工程と、

を含むアニメーション処理方法。

【請求項４】　請求項３に記載の方法において、

前記対応輪郭線検出工程は、隣接するキーフレーム間で輪郭線の数が一致しないとき、それらのキーフレーム間で輪郭線の自然消滅または自然発生があったものとみなし、その自然消滅または自然発生を考慮して輪郭線の対応関係を把握するアニメーション処理方法。

【請求項５】　請求項３に記載の方法において、

前記対応輪郭線検出工程は、隣接するキーフレーム間で輪郭線の自然消滅または自然発生があった場合、輪郭線の木構造をもとに自然消滅または自然発生すべき輪郭線を推定して輪郭線の対応関係を把握するアニメーション処理方法。

【請求項６】　請求項３に記載の方法において、

前記対応輪郭線検出工程は、アニメーション立体の連結成分の数に関する指定にもとづいて輪郭線の分岐を考慮し、輪郭線の対応関係を把握するアニメーション処理方法。

【請求項７】　請求項３に記載の方法において、

前記表面生成工程は、対応する輪郭線上において対応しあう点をそれらの点の近さをもとに検出し、これらの対応しあう点をホモトピーの軌跡で連続的につなぐことで立体の表面を生成するアニメーション処理方法。

【請求項８】　請求項７に記載の方法において、

2

前記表面生成工程は、対応する２つの輪郭線Ａ、Ｂそれぞれの上の点ａ、ｂについて、点ａから見て輪郭線Ｂ上の最も近い点が点ｂであり、かつ点ｂから見て輪郭線Ａ上の最も近い点が点ａであるとき、これら点ａ、ｂを対応しあう点として検出するアニメーション処理方法。

【請求項９】　請求項３に記載の方法において、

前記対応輪郭線検出工程は、隣接するキーフレームに描かれているオブジェクトの移動による影響を差し引いたうえで、それらキーフレーム上に存在する輪郭線どうしの近さを判断するアニメーション処理方法。

【請求項１０】　請求項３に記載の方法において、

前記対応輪郭線検出工程は、隣接するキーフレームに描かれているオブジェクトの輪郭線どうしの近さを把握した後、対応しあう確率が高い順に輪郭線の対を表示し、ユーザに対応関係を指定する機会を与えるアニメーション処理方法。

【請求項１１】　請求項３に記載の方法において、

前記対応輪郭線検出工程は、隣接するキーフレームに描かれているオブジェクトの輪郭線の一方が開曲線、他方が閉曲線の場合、前記閉曲線が前記開曲線の両端をつないだ状態に対応する場合にはユーザがその旨の指定をする機会を与えるアニメーション処理方法。

【請求項１２】　フレームをキーフレームとそれ以外の中間フレームに分類して処理を行うアニメーション処理方法において、

オブジェクトが描かれた複数のキーフレームを準備するキーフレーム準備工程と、

アニメーション立体を生成する工程であって、その立体の断面の輪郭線形状を前記キーフレームに描かれたオブジェクトの輪郭線形状に一致させたうえで、その立体の表面を前記断面の輪郭線どうしをつなぐホモトピーの概念を用いて生成する立体生成工程と、

を含むことを特徴とするアニメーション処理方法。

【請求項１３】　請求項１２に記載の方法において、

前記キーフレーム準備工程は、フレームに描かれるべきオブジェクトの二次元形状が時間の経過とともに位相同形でなくなるようなフレームをキーフレームとして準備するアニメーション処理方法。

【請求項１４】　請求項１２に記載の方法において、該方法はさらに、

生成されたアニメーション立体の前記断面とは異なる断面における輪郭線形状をもとに中間フレームに含まれるオブジェクトの二次元形状を取得する補間工程を含むことを特徴とするアニメーション処理方法。

【請求項１５】　請求項１２に記載の方法において、

前記アニメーション立体は、前記キーフレームの数と同じ数の互いに平行な断面によって定義される立体であって、それらの断面に垂直な軸がキーフレームの表示時刻を示す時間軸に対応するような三次元形状をもつ立体であるアニメーション処理方法。

【請求項１６】　フレームをキーフレームとそれ以外の中間フレームに分類してアニメーション処理を行うプログラムを格納した記録媒体であって、
そのプログラムは、
オブジェクトが描かれた複数のキーフレームの入力を受け付けるキーフレーム受付モジュールと、
アニメーション立体を生成するモジュールであって、その立体の断面の輪郭線形状を前記キーフレームに描かれたオブジェクトの輪郭線形状に一致させたうえで、その立体の位相に注目しながらその立体の表面を生成する立体生成モジュールと、
生成されたアニメーション立体の前記断面とは異なる断面における輪郭線形状をもとに中間フレームに含まれるオブジェクトの二次元形状を計算する補間モジュールと、
を含むことを特徴とする記録媒体。
【請求項１７】　フレームをキーフレームとそれ以外の中間フレームに分類して処理を行うアニメーション処理システムであって、
オブジェクトが描かれた複数のキーフレームの画像と、断面の輪郭線形状が前記キーフレームに描かれたオブジェクトの輪郭線形状に一致するようなアニメーション立体とを送信する送信装置と、
前記画像およびアニメーション立体を受信する受信装置と、
を含み、前記受信装置は前記アニメーション立体の前記断面とは異なる断面における輪郭線形状をもとに中間フレームに含まれるオブジェクトの二次元形状を取得することを特徴とするアニメーション処理システム。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】この発明は、アニメーションを処理する方法、特に、画像をフレームをキーフレームとそれ以外の中間フレームに分類して処理を行う方法に関する。この方法は、例えばアニメーション作成の際のインビットウィーニング（inbetweening）またはトゥウィーニング（tweening）と呼ばれる中割処理の自動化に利用することができる。この発明はまた、前記の方法を実現するプログラムを記録する記録媒体、および前記の方法を利用したアニメーションの送受信システムに関する。
【０００２】
【従来の技術】テレビ、ビデオ再生装置をはじめ、各種マルチメディア機器や画像通信技術の進歩により、今日の情報文化、娯楽文化において、映像は音声とともに中心的な役割を果たしている。各種映像に対する需要の拡大が映像技術の進展を促し、映像技術の進歩が新たな映像文化を産んでいる。アニメーションの世界でも、画像の高品質化やコンピュータ・グラフィックス（以下ＣＧ）技術の導入により、かつて想像もできなかった複雑



精緻な、または極めて芸術性の高い作品が製作されるようになった。
【０００３】今日のテレビの標準規格のひとつにＮＴＳＣがある。この規格では、表示フレーム数は通常３０ＦＰＳ（frames per second）である。また、リミテッド・アニメーションという簡易的な表示の場合は１５ＦＰＳ、映画の場合はこの数字が通常２４ＦＰＳとなる。したがって、テレビや映画などのアニメーションを作成するとき、最低でも１５ＦＰＳに相当するセル画を描く必要がある。１時間のアニメーションの場合、セル画は数万という膨大な枚数になる。
【０００４】こうしたセル画を効率的に制作する手法として、従来より中割手法が知られている。この手法は、まずアニメータがキーフレームと呼ばれる重要なフレームを描き、キーフレーム間のフレーム（以下「中間フレーム」という）を他の数名のアニメータが分業して描き足すというものである。ただし、描画自体は依然手作業によるため、効率化には限界がある。
【０００５】最近では、この中割手法がＣＧのアニメーション・プログラムにも取り入れられている。こうしたアニメーション・プログラムでは、時間軸上にオブジェクトの動きを割り付けていく。キーフレームはイベントの起こった箇所に設けられる。例えば金槌でくぎをたたくアニメーションを作る場合、キーフレームは２つ存在する。１番目のキーフレームは金槌を振り上げて構える場面、２番目は金槌がくぎに当たる場面である。したがって、まずこれらのキーフレームにおけるオブジェクトの画像をコンピュータに入力する。
【０００６】つづいて、２つのキーフレーム間のオブジェクトの位置を計算することによって中割が行われる。最も簡単な計算は線形補間であり、オブジェクトは２つのキーフレーム間を一定の速度で移動していく。最近では、キーフレーム間でオブジェクトの速度が変わるような場合を想定し、イーズ・イン（加速）やイーズ・アウト（減速）と呼ばれる手法も実現されている。また、オブジェクトの移動経路をスプライン曲線で表現することにより、より自然で滑らかな中間フレームを生成する技術も知られている。
【０００７】特開昭６０－１９１３６６号公報には、従来一般的な中割技術が開示されている。この公報では、キーフレームに描かれたキャラクタが複数の曲線で構成されるとき、それらの曲線の複数のキーフレーム間における対応関係を予め記述することより、中間フレームにおけるキャラクタの形状を自動的に算出、生成しようというものである。
【０００８】
【発明が解決しようとする課題】各種アニメーション・プログラムに中割機能をもたせることにより、アニメーション制作の効率を改善することができる。しかしながら、中割を行うための準備として、ユーザはキーフレー

ム間で対応しあう点を予め入力したり、対応しあう点が頂点になるような多角形を描画しなければならないなど、通常のフリーハンドの描画作業とは異なる作業を強いられる。このため、アニメータに負担が生じている。

【０００９】さらに大きな問題として、アニメーション・プログラムによって実際に中割を行うと、しばしば中間フレームにおけるオブジェクトが不合理な形状をとることがある。例えば、目の前で掌を１８０°回転させるようなアニメーションの場合、キーフレーム間で対応しあう指を自動認識することは容易ではない。したがって、あるキーフレームの中指と次のキーフレームの人差し指の間が間違って対応づけられるなど、現実にはありえない中間フレームが生成されるおそれがある。

【００１０】一般に、フレーム間のオブジェクトの正しい対応づけは画像認識の分野で以前から研究されており、エリアマッチング等の種々の手法が提案されている。それにも拘らず、高価なハードウエアと長い計算時間を許したとしても、完全な対応づけはほとんど不可能と考えられている。対応付けの精度を高めるためには、オブジェクトのキーフレーム間における動きが少なくなるよう非常に多数のキーフレームを設ける必要がある。しかし、これではアニメーション制作の効率化に逆行するし、それでも満足できる精度で対応づけがなされることは稀である。特に、掌の回転の具合によって複数の指が重なって見えたり離れて見える場面が混在する場合、画像認識によるアプローチはきわめて困難である。

【００１１】

【課題を解決するための手段】本発明の目的は、キーフレームから中間フレームを自動生成する際に中間フレームにおけるオブジェクトの形状が妥当なものになること、準備すべきキーフレームの数を減らすこと、フリーハンドによるキーフレームの描画を認めること、キーフレームに予めフレーム間の対応情報を与える必要をなくすこと、比較的簡素なハードウエアを用いて短い計算時間で中間フレームを生成すること、これら一連の処理に当たってユーザの操作性が簡単であることにある。

【００１２】これらの目的を達成するために、本発明者は、まずどのような場合に中間フレームにおけるオブジェクトの形状が不合理になりうるかを解析した。その結果、位相幾何学（トポロジー）の概念を導入することにより、この問題が説明できることを明らかにした。すなわち、オブジェクトの形状の妥当性はそのオブジェクトの位相という観点で捉えることができるのである。本発明者は、位相幾何学、特に後述のモース理論を拡張することにより、オブジェクトの形状の妥当性を確保するための方法を見いだした。

【００１３】本発明では、キーフレームにおけるオブジェクトの形状が、ある立体の断面の輪郭線形状に等しくなるような立体を想定する。この立体をアニメーション立体と呼ぶ。いったんアニメーション立体ができれば、



あとはこの立体の断面を切り出すことにより、中間フレームにおけるオブジェクトの形状を得ることができる。逆にいえば、ある断面の形状がその断面の位置に対応する中間フレームにおけるオブジェクトの形状に一致するような立体がアニメーション立体である。

【００１４】例えば第１のキーフレームに１つの球Ａ、第２のキーフレームには２つの球Ｂ、Ｃが描かれていたとする。このとき、

ケース１．球Ｂ、Ｃのうち一方（球Ｂとする）は球Ａそのものであり、他方（球Ｃとする）はいずれかの中間フレームで出現した、と考えることができる。この他に、

ケース２．球Ａが分裂して球Ｂ、Ｃになった、という可能性もある。もちろん、この他にもいくつかの可能性はある。

【００１５】しかしいずれの場合でも、アニメーション立体を導入することにより、位相幾何学の観点から説明することが可能である。すなわち、１．は「球ＡとＢは同じ連結成分に含まれ、球Ｃは別の連結成分に含まれる」と表現できる。連結成分とは、それぞれが個別の実体と考えてよい。つまり、アニメーション立体は球ＡとＢを断面として含む立体と、球Ｃを断面として含む立体の、合計２つの立体の集合として構成される。一方、２．は「球Ａ、Ｂ、Ｃはすべて同じ連結成分に含まれる」と表現できる。この場合、アニメーション立体は単一の立体で構成される。アニメーション立体は、球Ａを断面とする箇所から球ＢおよびＣを断面とする箇所の間で分岐する。分岐に関する考察も位相幾何学によって可能になる。

【００１６】このように、本発明では位相という概念を意識してアニメーション立体を生成するため、中間フレームにおけるオブジェクトの妥当な形状を条件に応じて提示することができる。例えば、一般則として「球は分裂しない」という条件が与えられれば、本発明では正しく上記１．の場合の中間フレームを生成する。分裂という現象は位相幾何学では分岐（ブランチ）として扱われる。オブジェクトの形状のとりうる可能性の明確な分類は、位相幾何学の助けを借りることによって可能となるのである。

【００１７】（１）具体的には、本発明のアニメーション処理方法は、フレームをキーフレームとそれ以外の中間フレームに分類して処理を行う。この方法は、オブジェクトが描かれた複数のキーフレームを準備するキーフレーム準備工程と、アニメーション立体を生成する工程であって、その立体の断面の輪郭線形状を前記キーフレームに描かれたオブジェクトの輪郭線形状に一致させたうえで、その立体の位相に注目しながらその立体の表面を生成する立体生成工程とを含む。

【００１８】この構成において、まず複数のキーフレームが準備される。つづいて、アニメーション立体を生成する。アニメーション立体の形状はその複数の断面の輪

郭線形状によって規定され、それら輪郭線形状はそれぞれ対応するキーフレームに描かれたオブジェクトの輪郭線形状に一致する。つづいて、アニメーション立体の位相に注目しながらその立体の表面を生成し、立体が完成する。

【００１９】いったんアニメーション立体が完成すれば、あとはこの断面を切り出すことで任意の中間フレームが得られる。このため、少ないキーフレームから多数の中間フレームを生成することができる。また、アニメーション立体の状態で保存することにより、アニメーションのデータベース化が可能になる。

【００２０】（２）本発明のある態様は、生成されたアニメーション立体の前記断面とは異なる断面における輪郭線形状をもとに中間フレームに含まれるオブジェクトの二次元形状を取得する補間工程を含む。この補間工程がいわゆる中割工程である。

【００２１】（３）本発明のある態様では、前記立体生成工程は、隣接するキーフレームに描かれているオブジェクトの輪郭線どうしの近さをもとに隣接するキーフレーム間における輪郭線の対応関係を把握する対応輪郭線検出工程と、対応する輪郭線間の表面をホモトピーの概念を用いて生成する表面生成工程とを含む。

【００２２】例えば前述のケース１．の場合、球Ａが球Ｂ、Ｃのいずれになったかを判定する際、球Ａと球Ｂ、球Ａと球Ｃ、それぞれの近さを計算する。近さは、例えば球Ａの輪郭線上に一定間隔で点をとり、これらの各点から球Ｂの輪郭線上の最も近い点までの距離を求めてこれらを合計し、この合計値の小ささで評価する。この合計値を球Ｃについても求め、例えば球Ｂに関する合計値のほうが小さいければ球Ａは球Ｂになったと考える。この作業を繰り返すことにより、異なるキーフレーム間で対応しあう輪郭線の対（以下「対応輪郭線対」という）が判明する。

【００２３】つづいて、対応輪郭線対の間の表面をホモトピーの概念で生成する。ホモトピーの正確な定義は後述するが、概略的には、ホモトピーはある関数ｆを別の関数ｇに連続的に変形していく関数Ｆである。したがって、対応輪郭線対の一方の形状関数をｆ、他方のそれをｇとしてホモトピーＦを定めれば、そのＦによって対応輪郭線対の間の表面が生成される。これをすべての対応輪郭線対の間で行うことにより、アニメーション立体が完成する。

【００２４】この態様によれば、異なるキーフレーム間で互いに近い位置にある輪郭線どうしが対応するであろうという経験則をもとに、輪郭線の対応関係の把握を自動化することができる。このため、ユーザはキーフレームをフリーハンドで描画することができ、わざわざキーフレーム間の画像の対応関係を入力する必要もない。中間フレームにおけるオブジェクトの形状も妥当なものになる。

【００２５】また、この態様によれば、ホモトピーの概念によって立体の表面が生成されるため、このホモトピーの関数形を決めておけば表面の一意的な表現が可能となる。立体のモデリングの際に表面パッチの貼り付けを行う場合、通常その貼り付け方が一意的には決まらず、したがって自動化が困難とされる。本発明は副次的な効果としてアニメーション立体の一意的表現が可能になり、したがってアニメーション自体の一意的な表現が可能になる。

【００２６】（４）本発明のある態様では、前記対応輪郭線検出工程は、隣接するキーフレーム間で輪郭線の数が一致しないとき、それらのキーフレーム間で輪郭線の自然消滅または自然発生があったものとみなし、その自然消滅または自然発生を考慮して輪郭線の対応関係を把握する。ここで「自然消滅」「自然発生」とは、分岐によって発生または消滅するのではなく、それ自体、他と無関係に発生または消滅することをいう。

【００２７】この態様は、上述の球の例においてケース１を選択することに等しい。すなわち、ケース１では球Ａの輪郭線は球Ｂのそれに連続的につながっていくが、球Ｃの輪郭線はいずれかの中間フレームで自然発生する。一方、ケース２では球Ａの輪郭線が球ＢおよびＣに分岐するだけで、新たな輪郭線が自然発生するわけではない。

【００２８】この態様は「実存する三次元のオブジェクトが分裂したり融合する可能性は低い」という経験則に基づいている。球の例でも、ケース２よりはケース１のほうがより一般的な現象である。したがって、本態様のような決め方をすることにより、ユーザがケースを指定しなくても、大抵の場合に所望のアニメーションを得ることができる。この結果、ユーザの操作はより簡単になる。

【００２９】（５）本発明のある態様では、前記対応輪郭線検出工程は、隣接するキーフレーム間で輪郭線の自然消滅または自然発生があった場合、輪郭線の木構造をもとに自然消滅または自然発生すべき輪郭線を推定して輪郭線の対応関係を把握する。

【００３０】例えば、トーラス（ドーナツ）が９０°回転したとき、最初に中心の穴が見えていたとしても、これが見えなくなる場合がある。輪郭線の中に別の輪郭線があるとき、前者を親輪郭線、後者を子輪郭線と定義すれば、トーラスの外側の円は親輪郭線、内側の円は子輪郭線となる。したがって、消えるのは子輪郭線のほうである。通常の三次元オブジェクトでは、子輪郭線は親輪郭線の存在を前提に存在するため、子輪郭線だけが残って親輪郭線が消えるという現象は起こらない。同様に、自然発生する輪郭線が既存の輪郭線の親輪郭線になることもない。本態様では、こうした観点から自然消滅または自然発生する輪郭線を推定する。「木構造」とは輪郭線の親子関係を示すもので、前記推定を可能ならしめる

ものである。

【００３１】本態様によれば、ユーザが自然消滅等する輪郭線をわざわざ指定する必要がなく、ユーザの操作がより容易になる。

【００３２】（６）本発明の別の態様では、前記対応輪郭線検出工程は、アニメーション立体の連結成分の数に関する指定にもとづいて輪郭線の分岐を考慮し、輪郭線の対応関係を把握する。

【００３３】球の例において、ケース１で関係する連結成分は２であり、輪郭線の数は１から２に変化し、輪郭線は分岐しない。このため、「輪郭線の数が連結成分の数以下のときは輪郭線の分岐はない」と一般化することができる。一方、ケース２で関係する連結成分は１であり、輪郭線の数はケース同様に１から２に変化する。ケース２では輪郭線は分岐している。このため、「連結成分よりも輪郭線が多いときは輪郭線の分岐がある」と一般化することができる。

【００３４】このように、予め隣接するキーフレーム間で関係する連結成分の数が与えられれば、あとは輪郭線の数から自動的に輪郭線の分岐の有無を判定することができる。

【００３５】（７）本発明のある態様では、前記表面生成工程は、対応する輪郭線上において対応しあう点（以下「対応点対」ともいう）をそれらの点の近さをもとに検出し、これらの対応点対をホモトピーの軌跡で連続的につなぐことで立体の表面を生成する。

【００３６】本態様では、対応点対を自動的に決めることができるため、アニメータがキーフレーム間の対応点対をわざわざ指定したり入力する必要がない。

【００３７】（８）本発明のある態様では、（７）のとき前記表面生成工程は、対応する２つの輪郭線Ａ、Ｂそれぞれの上の点ａ、ｂについて、点ａから見て輪郭線Ｂ上の最も近い点が点ｂであり、かつ点ｂから見て輪郭線Ａ上の最も近い点が点ａであるとき、これら点ａ、ｂを対応点対として検出する。以降、これらの点を「最近点対」ともよぶ。

【００３８】この態様は後述の連続トロイダルグラフを用いた対応点対検出の基本原理に当たる。この方法によれば、輪郭線Ａ、Ｂ上の対応点対がそれぞれの輪郭線全周を見渡したうえで決まるため、従来の離散トロイダルグラフ（後述）に比べてより自然な対応関係を見い出すことができる。

【００３９】（９）本発明のある態様では、前記対応輪郭線検出工程は、隣接するキーフレームに描かれているオブジェクトの移動による影響を差し引いたうえで、それらキーフレーム上に存在する輪郭線どうしの近さを判断する。

【００４０】例えば、カメラでオブジェクトを撮影するような場合、カメラの移動によってすべてのオブジェクトが移動する。この状態でキーフレーム間の対応輪郭線対を検出すると、誤対応が増えると考えられる。カメラが動かなくとも、複数のオブジェクトが右から左に移動するような場合に同様の問題が起こる。そこでこの態様では、移動の影響を差し引いたうえで輪郭線どうしの近さを判断する。具体的には、すべてのオブジェクトをもとの位置に戻したうえで近さを判定するなどの方法がある。

【００４１】この態様によれば、対応輪郭線対の検出精度が高まるため、準備すべきキーフレームの数をさらに減らすことができる。

【００４２】（１０）本発明のある態様では、前記対応輪郭線検出工程は、隣接するキーフレームに描かれているオブジェクトの輪郭線どうしの近さを把握した後、対応しあう確率が高い順に輪郭線の対を表示し、ユーザに対応関係を指定する機会を与える。

【００４３】この態様によれば、輪郭線対を検出する際に生じうる誤対応の問題を容易に解消することができる。ユーザの指定操作も簡単である。

【００４４】（１１）本発明のある態様では、前記対応輪郭線検出工程は、隣接するキーフレームに描かれているオブジェクトの輪郭線の一方が開曲線、他方が閉曲線の場合、前記閉曲線が前記開曲線の両端をつないだ状態に対応する場合にはユーザがその旨の指定をする機会を与える。開曲線とは、有限の長さをもつ曲線であって閉曲線でないもの、すなわち曲線セグメントをいう。

【００４５】対応する輪郭線の一方が開曲線、他方が閉曲線の場合、閉曲線が開曲線の両端をつないだ状態に対応する場合のほか、例えば閉じている目が開くように、両端が固定されたまま中央部分が２つに開いて分かれた状態に対応する場合がある。そこで、本発明ではユーザがそれらのいずれを望むかを指定できるものとする。

【００４６】（１２）本発明の別の態様は、オブジェクトが描かれた複数のキーフレームを準備するキーフレーム準備工程と、アニメーション立体を生成する工程であって、その立体の断面の輪郭線形状を前記キーフレームに描かれたオブジェクトの輪郭線形状に一致させたうえで、その立体の表面を前記断面の輪郭線どうしをつなぐホモトピーの概念を用いて生成する立体生成工程とを含む。

【００４７】ここでは（１）同様にまず複数のキーフレームが準される。つづいて、立体生成工程でアニメーション立体が生成される。アニメーション立体の表面はホモトピーの概念を用いて生成される。この結果、いつでも中割が可能な状態が実現される。

【００４８】（１３）本発明のある態様では、（１２）のキーフレーム準備工程は、フレームに描かれるべきオブジェクトの二次元形状が時間の経過とともに位相同形でなくなるようなフレーム（以下「臨界フレーム」ともいう）をキーフレームとして準備する。臨界フレームの例は、１つの物体が引き延ばされて２つの物体に分裂す

るとき、その分裂する瞬間のフレームである。本明細書では、アニメーション立体の位相が変わることと臨界フレームが存在することは同義と考えてよい。なぜなら、アニメーション立体の臨界フレームに相当する断面において、１つの物体が２つの物体に分岐等するためである。

【００４９】この態様では、実際にオブジェクトの位相が変化する場面でその変化がキーフレームに反映される。したがって、中間フレームでは逆に位相の変化がないことになり、ホモトピーによる表面の生成に好都合である。なぜなら、ホモトピー自体は連続関数であり、ホモトピーの軌跡して生成された表面はアニメーション立体の位相を変化させないためである。

【００５０】この態様によれば、キーフレームから中間フレームを生成する際、中間フレームにおけるオブジェクトの形状が妥当なものになる確率が高まる。なぜなら、位相の変化はキーフレームによって指定でき、一方、キーフレーム以外の位相の変化しないフレームは位相を変化させないホモトピーの概念を用いて生成されるためである。この態様では、システムが位相の変化を推定する必要はない。実際のアニメーション作成現場でも、アニメータは無意識のうちに臨界フレームをキーフレームにしていることが多く、インプリメンテーションの面でも無理がない。

【００５１】（１４）（１２）において本発明のある態様は、生成されたアニメーション立体の前記断面とは異なる断面における輪郭線形状をもとに中間フレームに含まれるオブジェクトの二次元形状を取得する補間工程を含む。このため、（２）同様、容易に中割が実現される。

【００５２】（１５）（１２）において本発明のある態様では、前記アニメーション立体は、前記キーフレームの数と同じ数の互いに平行な断面によって定義される立体であって、それらの断面に垂直な軸がキーフレームの表示時刻を示す時間軸に対応するような三次元形状をもつ立体である。すなわち、アニメーション立体の断面をｘｙ平面に平行におくとき、ｚ軸が時間軸となる。

【００５３】仮に２つのキーフレームの表示時刻をｔ＝ｔ０、ｔ＝ｔ１とすると、これら２つのキーフレームのちょうど中間にある中間フレームの表示時刻はｔ＝（ｔ０＋ｔ１）／２である。２つのキーフレームはそれぞれアニメーション立体のｚ＝ｔ０、ｚ＝ｔ１における断面である。また、前記中間フレームは同様にｚ＝（ｔ０＋ｔ１）／２における断面となる。アニメーション立体をこのように定義することにより、中割処理を少ない計算量で容易に行うことができる。

【００５４】（１６）一方、本発明のさらに別の態様は記録媒体に関する。この記憶媒体にはアニメーション処理を行うプログラムが格納される。このプログラムは、オブジェクトが描かれた複数のキーフレームの入力を受け付けるキーフレーム受付モジュールと、アニメーション立体を生成するモジュールであって、その立体の断面の輪郭線形状を前記キーフレームに描かれたオブジェクトの輪郭線形状に一致させたうえで、その立体の位相に注目しながらその立体の表面を生成する立体生成モジュールと、生成されたアニメーション立体の前記断面とは異なる断面における輪郭線形状をもとに中間フレームに含まれるオブジェクトの二次元形状を計算する補間モジュールとを含む。このプログラムは（１）および（２）の処理を行うものと考えられる。

【００５５】（１７）本発明のさらに別の態様はアニメーションを通信するシステムに関する。このシステムは、オブジェクトが描かれた複数のキーフレームの画像と、断面の輪郭線形状が前記キーフレームに描かれたオブジェクトの輪郭線形状に一致するようなアニメーション立体とを送信する送信装置と、前記画像およびアニメーション立体を受信する受信装置とを含む。

【００５６】この構成において、受信装置がアニメーション立体を受信した後、アニメーション立体の断面を切り出すことにより、その断面における輪郭線形状をもとに中間フレームに含まれるオブジェクトの二次元形状を取得する。ある程度多数の中間フレームを取得すれば、これらを順次表示することにより、アニメーションを再生することができる。

【００５７】このシステムによれば、非常に少ない通信データ量でアニメーションを送ることができる。

【００５８】

【発明の実施の形態】本発明の好適な実施形態を説明する。本発明を理解するに際し、本発明者が先に公表した論文の内容を「前提技術」として説明することは有用である。この前提技術は本発明者のひとりの論文（東京大学博士論文１９９３年品川嘉久）の一部である。

【００５９】前提技術は、位相幾何学の考え方を導入して立体を正確に表現するための技術である。一方、本発明は立体の三次元形状ではなく、その断面の二次元形状を最終目的とし、立体を異なる意味合いで用いる。本発明では、立体の断面の形状からアニメーションの各フレーム画像を得る。以下、前提技術を引用した後、その前提技術に対して必要な修正または拡張を加える形で実施形態を説明する。

【００６０】［前提技術］

［１］モース理論に基づく曲面符号化システム

（１）はじめに

モース（Morse）理論を数学的ツールとして三次元物体の解釈を行う。しかし、三次元の曲面を完全な正確さをもって符号化するためにはモース理論だけでは不十分である。以下、その理由を説明し、モース理論を拡張することによってこの問題の解消を図る。

【００６１】三次元空間における立体や曲面の形状を符号化するとき、通常これらをなんらかの記号の配列とし

て表現する。自然界に見られるオブジェクトの場合は、形状は非常に多くの自由度をもつ。そのため符号化の際には一定の単純化が必要となる。位相幾何学（トポロジー）はこうした単純化を行うための数学的手段である。

【００６２】（２）符号化システムに対する要請

自然界の物体の形状データは、しばしば断面という形で利用可能である。例えばＸ線断層撮影などである。このデータを符号化に用いれば、隣接する断面の輪郭線間で三角パッチなどの内挿を行う曲面再構成システムによる利用が可能になる。以下説明する符号化方法も断面の輪郭線（以下単に「輪郭線」とも呼ぶ）に基づく。

【００６３】符号化システムは、例えば人の内耳の半規管のように、分岐、ハンドル（穴）または多層曲面からなる内部構造をもつような物体も符号化できなければならない。こうした物体を符号化するために、一連の断面の輪郭線を記録していくだけでは、対象である曲面を十分に説明することにはならない。

【００６４】符号化システムでは、符号化された曲面の位相的な正しさ（トポロジカル・インテグリティ）が維持される必要がある。そうでないと、符号化された曲面が物理的に意味をなさないことがある。図１は輪郭線構造の不当な変換例を示している。ここでは２つの輪郭線があり、一方が他方に包含されている。したがって、本来前者は後者の境界線を超えることはできず、この規則が守られない限り正しい符号化はできない。曲面符号化システムには位相的な正しさの維持が要求されるのである。

【００６５】（３）古典的なモース理論

もともとモース理論は変分法を取扱うために提唱された。そしてその目的は、無限次元の道の空間における汎関数の極小値を記述することにあった。このことから逆に、汎関数の極小値を利用することにより、それ以外の方法での記述が困難であるような空間の位相的な特徴を記述することが可能になる。以下、モース理論の概要を説明する。

【００６６】◎ 可微分多様体

モース理論を適用できる空間は可微分多様体である。有限次元の多様体を考えてみる。

【００６７】いま任意の整数ｎについて、ｎ次元多様体は位相空間であって、そこではすべての点がｎ次元空間$R^n$の部分集合の上に一対一かつ両連続に写像可能な近傍をもつとする。このような写像は「チャート」と呼ばれ、その領域に含まれる点について局所座標系を提供する。地球を例にとれば、緯度と経度が局所座標系に当たる。多様体がｐ回微分可能であるためには、一方の座標系から他方の座標系への変換が、２つの異なるチャートの値域に含まれる点についてｐ回微分可能でなければならない。

【００６８】このため多様体は、可微分的に重なり合った$R^n$の領域から構成されていると考えることができる。例えば、直線や円周は一次元多様体の構造を与えられる。また、球の表面は例えば北半球と南半球のように、少なくとも２つのチャートを用いた二次元多様体を用いて表現できる。同様にトーラスの表面は、少なくとも４つのチャートを用いた二次元多様体によって表現できる。$R^3$から結び目のある円を取り除けば三次元多様体の一例となるし、さらに高次の多様体は例えばロボットの腕のコンフィグレーション空間として現れる。

【００６９】◎ 可微分写像および特異点

チャートを用いることにより、ｐ次元多様体からｎ次元多様体への写像は$R^p$の各区分から$R^n$の各区分への写像として（区分ごとに）数値表現することができる。これらの写像については微分可能性を検証することができる。要素がｋ回連続的に微分可能であれば、その写像は$C^k$級である。

【００７０】ここで局所座標系の上で高さ関数を定義する。高さ関数は与えられた点の高さ（物体が埋め込まれている三次元空間におけるｚ座標など）を返す関数である。

高さ関数ｈ：$R^2 \rightarrow R$のヤコビ行列は、

【数１】

$$J = \begin{pmatrix} \frac{\partial h}{\partial x_1} \\ \frac{\partial h}{\partial x_2} \end{pmatrix}$$

で与えられる。ヤコビ行列は各点について計算することができ、そのランクの最大値はｎとｐの小さいほうである。ヤコビ行列のランクがこの最大値に等しい点は「正則点」と呼ばれ、それ以外の点は「特異点（singular point）」または「臨界点（critical point）」と呼ばれる。例えば、高さ関数に関する特異点には、頂上（peak）、鞍点（saddle point）、谷底点（pit）がある。別のいいかたをすれば、特異点はヤコビ行列がゼロベクトルとなる点であり、特異点では法線ベクトルが高さ方向と同じ方向を向く。なお、特異点が写像される点は特異値と呼ばれる。

【００７１】◎ ヘッセ行列と指数

ｎ次元多様体からＲへの写像（以下「多様体上の関数」と呼ぶ）については、ある点における偏微分の値がすべて０の場合、その点が臨界点となる。そのような点において、前述の関数は二次偏微分に基づく二次形式によって近似される。この行列表示はヘッセ行列と呼ばれ、その要素は以下のように記述される。

【００７２】

【数２】

15　　16

$$h = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_1 \partial x_2} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{pmatrix}$$

特異点におけるヘッセ行列の負の固有値の数をその特異点の指数（index）と呼ぶ。指数は後述の図２に示すとおり、被約形式におけるマイナス符号の数に等しい。すなわち、頂上の指数は２、鞍点は１、谷底点は０である。後述の図４（ａ）～（ｃ）のごとく、トーラスの場合、指数２、１、０の臨界点はそれぞれ１個、２個、１個である。

【００７３】臨界点におけるヘッセ行列のランクがｎであるとき、その特異点は縮退していない、すなわち「非縮退」と呼ばれる。どのようなC²級関数でも、モース関数によって近似することができる。モース関数とは、臨界点の縮退がないような関数をいう。したがって、モース関数の臨界点は孤立しているはずであり、コンパクトな多様体に関する限り、臨界点は有限個しか存在しない。いずれの２つの特異値も等しくないと仮定することは非常に好都合であり、その仮定の下でも同じ近似結果が得られた。

【００７４】◎ ホモトピーの型

モース理論によれば、多様体とその多様体上のモース関数が与えられ、その関数の特異点の指数の列び方が判明すれば、各特異点に対応する一連の演算を行うことによって、その多様体と同じホモトピー型をもつ位相空間を胞複体（cell complex）として構築することができる。

【００７５】高さを表す任意の実数について、セル（胞体）はその実数の示す高さ以下の点からなる多様体の部分に関するモデルを与える。Ｒが上下に走査されるとき、２つの連続する特異値の間ではセルの位相は変化しないが、特異値を横切るたびに、それ以前のセルに対して「ｋ次元セル」をつなげていくことにより、胞セルを作り上げていくことができる。ここでｋは横切った特異点の指数である。簡単にいえば、物体の形はその臨界点の指数と同じ次元をもつセルというものを貼り合わせることで復元できる。

【００７６】図２は、特異点の指数、ｋ次元セルおよびそれによって符号化される物体の関係を示す図である。ここでは物体としてトーラスを挙げている。同図のごとく、注目する高さが臨界点を含む高さを横切るとき、その高さより下の点によって構成される位相が変化する。この変化は、位相的に見れば後述するようにｋ次元セルをつないでいくことと同じである。同図のごとく、二次元セル（ｋ＝２）はお椀を伏せたような形、一次元セル（ｋ＝１）は紐のような形、０次元セル（ｋ＝０）はひとつの点で表すことができる。もとの物体の形は、それらのセルをつなげた上で、粘土細工のように変形することで得られる。トーラスの場合、二次元セルを１個、一次元セルを２個、０次元セルを１個つなげて得られる。

【００７７】ここで注意すべきは、指数の配列だけではセルを完全に記述することはできないことである。図３（ａ）～（ｃ）は、それぞれが同じモースの指数の配列をもつ３組の曲面を示す。このように、指数の配列だけでセルを完全に決めることはできない。そのため、セルをつないでいくときにいずれの連結成分（それぞれ独立した実体）が関連するかを知らなければならない。

【００７８】レーブ（G.Reeb）は、多様体から位相商空間として得られるグラフを提唱した。レーブグラフは特異点の相互関係を示すもので、物体表面を等高線で表し、各等高線の連結成分をひとつの点として表すことで得られる。レーブは、多様体（コンパクトとする）において、モース関数の下で同じ値をもち、かつ対応する断面として同じ連結成分に含まれるすべての点を等化することにより、このグラフを導出した。つまり、２つの臨界点を含む平面間に存在する多様体の部分の連結成分はグラフの辺として表現され、各特異点はグラフの各頂点に対応する。レーブグラフは物体の骨格を示すグラフということができる。

【００７９】図４（ａ）～（ｃ）は、トーラスとそのレーブグラフの関係を示す図である。図４（ａ）は、もとのトーラス、図４（ｂ）はその断面図、図４（ｃ）はレーブグラフを示している。図４（ｂ）において、同一平面内にあって重なり合わない円の部分が、（ｃ）における２つの別々の辺に相当する。このレーブグラフはアイコンとして極めて表現力に優れているため、以降必要に応じてこのグラフをアイコン表示に用いる。

【００８０】（４）理論上の限界

重要なのは、モース理論をこのように古典的な方法で用いた場合、多様体に内在する位相的な性質を発見することができるに過ぎないことである。指数の配列だけでは、多様体が空間に埋め込まれている状態を符号化することができない。例えば、空間に埋め込まれたトーラスに結び目があるかどうかは知ることができない。図４（ｂ）に示すとおり、２つの異なる形状が同じ特異点に帰着するためである。同様に図４（ｃ）に示すとおり、連結があるかどうかもモース理論による単純な符号化では示せない。

【００８１】（５）モース理論に基づく符号化の拡張

ここでは、議論の対象をC²級（三次元空間に埋め込まれたC²）のコンパクトな二次元多様体の表面に限る。我々が用いる曲面上のモース関数は、空間における高さ関数から誘導する。事実、C²曲面をわずかに回転させれば臨界点の縮退をなくすことができるため、その高さ関数をモース関数にすることができる。

【００８２】モース理論によれば、２つの臨界レベル（臨界点が含まれる平面の高さ）の間では断面の位相は変化しない。このことから、曲り具合の異なる多くの円

筒を用いて、２つの臨界レベルの間の曲面をモデル化することができる。特異点のない断面は平面に埋め込まれた円から構成されるため、包含関係を表示するために構造的な符号化が必要となる。すなわち同じ円に含まれる複数の円をグループ化する符号化である。

【００８３】我々は、レーブグラフから得られる情報（すなわち各頂点間の接続状態）の他に、モースの指数に新たな情報を付け加える拡張符号化を提案する。すなわちこの情報は、２つの連続する臨界値の間で複数の円筒がどのように交換され、どのような向きに接続されるかに関する情報である。

【００８４】（６）符号化システムの例

符号化システムの概要を説明する。このシステムでは、曲面に対してｋ次元セルを次々につないでいくことにより曲面を表現する。そして各断面における輪郭線の階層構造の変化を追跡する。ここで、セルの接続を示す演算子を導入し、これらの演算子を用いて曲面を符号化する。演算子によって接続されていくセルをアイコンで表現することにより、符号化の対象となる曲面の構造の理解を容易にする。

【００８５】この符号化システムの最大の特徴は、得られる符号化の結果が位相の正しさを保証することにある。図１に示したとおり、断面間における輪郭線の階層構造の変化が物理的に意味のある曲面を規定するとは限らない。つまり、各断面における輪郭線の階層構造を符号化するだけでは、位相の正しさを保証することができない。したがって、輪郭線の階層構造を変化させるような演算を符号化することが必要になる。

【００８６】◎ 親子関係と輪郭線

輪郭線の階層構造の表現方法を説明する。ここでは木構造を用いる。

【００８７】まず、ある輪郭線が別の輪郭線を包含するとき、前者を後者の親輪郭線、後者を前者の子輪郭線と呼ぶ。図５（ａ）と（ｂ）は輪郭線の親子関係、およびその木構造による表現を示す図である。図５（ａ）のごとく、親子関係はネスティング構造にすることができる。以下、輪郭線の一番目を #1 で示す。 #1 は #2 の親輪郭線、 #2 は #4 の親輪郭線である。#1 や #7 のように親輪郭線をもたない輪郭線は「仮想輪郭線 #0」の子輪郭線と表現する。したがって、#0 は木構造の頂点にくる。

【００８８】ある輪郭線の親を示すために、配列 Parent#[] を定義する。たとえば Parent#[1]＝0 である。一方、ある輪郭線の子輪郭線は Children という配列にリストされ、その配列へのポインタが付される。例えば #3 の子輪郭線である #5 および #6 は、以下のように記述される。

【００８９】

Children[3]↑[1]＝5、 Children[3]↑[2]＝6

同じ親輪郭線をもつ輪郭線同士は兄弟輪郭線と呼ばれる。図５（ａ）の場合、#2 と #3 が兄弟輪郭線である。親輪郭線の親は祖父輪郭線、子輪郭線の子は孫輪郭線と呼ばれる。また、内部に物体が存在する輪郭線を中実輪郭線、存在しない輪郭線を中空輪郭線と呼ぶ。図５（ａ）の場合、#1、#4、#5、#6、#7 は中実輪郭線、#2 および #3 は中空輪郭線である。

【００９０】◎ セルを接続するための演算子

４つの演算子、Put_e0、Put_e1_merge、Put_e1_divide、Put_e2 を定義する。これらがセルを貼り付ける演算子である。以下、ｋ次元セルを $e^k$ と表示する。

【００９１】物体の構成は頂上から谷底に向けて進む。処理は、それ以上セルを接続することができなくなった時点で終了する。演算子によって構成しようとする曲面の状態を示すために、各断面における輪郭線を用いる。

【００９２】図６は、これらの演算子を用いてトーラスを構成する方法を示す。以下この図を用いて演算子の機能を説明する。

【００９３】１．同図の一番上に示すとおり、$e^2$ を生成するために Put_e2 (0) を実行する。このパラメータ「0」は #1 が #0 の内側に生成されることを示す。このセルの断面は同図の「断面表示」の箇所に示される。図のように、Put_e2 は断面の平面上に輪郭線を生成する機能をもつ。演算子によって生成される輪郭線に生成順の数字を与えるため、Put_e2 (0) によって生成された輪郭線は #1 である。

【００９４】新たに生成された輪郭線の状態は、初期値として常に「イネーブル」である。イネーブルとは、その輪郭線に対してセルを接続することが許される状態を示す。$e^2$ のアイコン表示を同図「アイコン」の下に示す。

【００９５】２．つづいて、Put_e1_divide (1,nil,inside) により、$e^2$ に対して $e^1$ を貼り付ける。新たに生成された輪郭線を #2 とする。パラメータ「inside」は #2が Parent#[1]＝0 の子輪郭線として生成されることを示す。二番目のパラメータは参照すべき子輪郭線のリストを示す。ここでは２番目のパラメータが「nil」であり、ここでは子輪郭線に対する操作、具体的には子輪郭線の削除はない。

【００９６】３．つぎに、Put_e1_merge (1,2) を用いて別の $e^1$ を貼り付け、#1 と #2 をマージする。この演算子は、１番目、２番目のパラメータによって示される輪郭線を１番目のパラメータの側にマージする。マージによって２番目のパラメータの示す輪郭線は消滅するため、それがその親輪郭線のもつ子輪郭線のリストから削除される。同時に、その輪郭線の状態が「イネーブル」から「ディセーブル」に変更される。したがって、この輪郭線に新たにセルを接続することができない。

【００９７】４．最後に Put_e0 (1) を用いて $e^0$ を貼り付け、#1 を閉じる。#1 の状態はイネーブルからディセーブルに変更される。アイコンがこの変更を反映して

いる。ある輪郭線にセル$e^0$が接続されたとき、その輪郭線のもつすべての子輪郭線が予めディセーブルされていなければならない。

【００９８】以上の手順により、イネーブルの状態で残っている輪郭線がなくなるため、演算子によるセルの貼り付けは完了する。

【００９９】図７～９は疑似パスカルコードによって演算子のプログラミング例を示す図である。図において、Max_children と Max_contour_number は十分大きな正の整数であり、メモリーアロケーションのためだけに用いられる。

【０１００】図７において、Number_of_children は各輪郭線の子輪郭線の数、Most_recently_created# は、最も最近生成された輪郭線の番号、Contour_status は、各輪郭線がイネーブルであるかディセーブルであるかを示す。Children はある輪郭線の子輪郭線のリストを保持する Child_list という配列に対するポインタである。Child_list の終了は定数 End_of_list によって示される。これらの変数は以下の手順で初期化される。

【０１０１】

```
most_recently_created#:=0;
  for i:=0 to max_contour_number do
      number_of_children[i]:=0;
  for i:=1 to max_contour_number do
      contour_status[i]:=disabled;
contour_status[0]:=enabled;
```

これにより、仮想輪郭線がイネーブルされる。つぎの create_new_contour は新しい輪郭線を作り、Most_recently_created# を増加させ、その状態を初期化する。

【０１０２】

```
procedure create_new_contour;
begin
  most_recently_created#:=most_recently_created#+1;
  contour_status[most_recently_created#]:=enabled;
end
```

図８において、後に使用するために２つのプロシージャと３つの関数を定義する。Add_listed_children は第２パラメータである clist にリストされている輪郭線を第１パラメータの子輪郭線（children [n]↑）のリストに追加する。Remove_listed_children は第１パラメータの子輪郭線（children [n]↑）のリストから第２パラメータである clist にリストされている輪郭線を削除する。これらのプロシージャはまた、Number_of_children および Parent#の配列を更新する。

【０１０３】一方、関数 Add_children (n, clist) は clist のすべての輪郭線が #n の子輪郭線であるとき「真」を返す。それ以外の場合「偽」を返す。関数 in_list (n, clist) は clist が #n を含むとき「真」を返し、それ以外の場合「偽」を返す。関数 List_containing_only (n) は２つプロシージャ、Add_listed_children および Remove_Listed_children に対して与えるべき輪郭線を１つだけ含むリストを作るために定義される。

【０１０４】これらにより、４つの演算子 Put_e2、Put_e0、Put_e1_divide および Put_e1_mergeを定義することができる。これらは図９に示される。

【０１０５】１．Put_e2 (n) は #n の子輪郭線として新たな輪郭線を生成する。

【０１０６】２．Put_e0 (n) は$e^0$を貼り付けることにより、#n を削除する。ここで、All_successors_disabled (n, contour_number) は、#n のすべての子輪郭線がディセーブルであるときに限り「真」を返す。

【０１０７】３．Put_e1_divide (n, clist, inside) は、#n を分割して新たな輪郭線を作る。Clist にリストされている輪郭線は新たに生成される輪郭線の子輪郭線になる。それらは、#r＝#n または #r＝parent#[n] の一方が成り立つとき、#r の子輪郭線のリストから削除される。（これは #n または parent#[n]のいずれの子輪郭線であったかに依存する）。これら２つの場合のみが許される。

【０１０８】４．Put_e1_merge (c1,c2)は #c2 を #c1 にマージする。#c2 は Parent#[c2]の子輪郭線のリストから削除される。#c2 のすべての子輪郭線は Parent#[c1]または #c1 の子輪郭線になる（そのいずれであるかは、#c1 が #c2 の親輪郭線であるか兄弟輪郭線であるかに依存する）。これら２つの場合のみが許される。

【０１０９】これらの演算子によって符号化される曲面の構造を容易に理解するために、曲面のレーブグラフを構成するセルの視覚的表示を提案する。

【０１１０】図１０に示すアイコンはそれぞれセルを表している。セルの貼り合わせは、一方のセルの平らな頂面と他方のセルの平らな底面を接触させることによって行われる。中空輪郭線には白いアイコン、中実輪郭線には黒いアイコンを用いる。$e^1$に関しては複数のアイコンが存在する。

【０１１１】図１１はセルの貼り合わせを示す図であり、同図に示すごとく子輪郭線のアイコンはその親輪郭線のアイコンの内側に描かれる。鉛直な軸に関する鏡像のアイコンも認められる。

【０１１２】アイコンの特徴は、それが貼り付けられる輪郭線の構造を維持する点にある。例えば図１１の上半分に示されるように、$e^1$がセルに接続されるとき、アイコンが接続されることになるセルの親子関係の構造が

維持される。

【０１１３】図１１の下半分に示すように、セルの高さの調整が必要な場合にはダミーアイコンを挿入する。レーブグラフは平面グラフではないため、ダミーアイコンを用いてセルを交差させることもできる。この結果、$e^1$を離れたセルに接続することが可能となる。$e^1$がセルを交換するとき、同時にその内部構造も交換する。階層的な輪郭線の構造を保持するために、ダミーアイコンはそれが貼り付けられる輪郭線の親輪郭線の境界を越えることはできないし、他の輪郭線に侵入することもできない。この理由から、ダミーアイコンを用いて交換することのできる輪郭線は兄弟輪郭線に限られる。

【０１１４】以上のまとめとして、図１２～１４に演算子を用いた物体の符号化の例を示す。図１２は、対象となる物体のレーブグラフをアイコンによって示したもの、図１３はその物体の断面の輪郭線を示したもの、図１４はその物体を構成するための演算子である。

【０１１５】［２］演算子を用いた符号からの曲面の構成

（１）ホモトピーの軌跡としての曲面の生成

前述の方法によって得られた符号化データをもとに物体の曲面を構成する。

【０１１６】すでに述べたとおり、臨界点を含む断面どうしの間では輪郭線の位相は変化しない。頂点から底辺まで走査したとき、輪郭線の形状は変化する。この輪郭線の変形はホモトピーを用いてうまく表現することができる。ホモトピーはある関数を他の関数に変換する。以下の説明においては、すべての輪郭線は形状関数によって表され、変形はホモトピーによって表されるとする。ホモトピーの定義は以下のとおりである。

【０１１７】〔定義〕Ｘ、Ｙが位相空間であるとき、ｆ、ｇ：Ｘ→Ｙという写像を考える。ここで、ｘ∈Ｘなるすべての点ｘに対して、

Ｆ(x,0)＝ｆ(x)

Ｆ(x,1)＝ｇ(x)

が成り立つような写像Ｆ：Ｘ×Ｉ→Ｙが存在する場合、「ｆとｇはホモトープである」といわれる。ここでＩ＝[0,1]∈Ｒである。またこのとき、写像Ｆは「ｆからｇへのホモトピー」と呼ばれる。Ｆが、

Ｆ(x,t)=(1−t)ｆ(x)＋tｇ(x)

で定義されるとき、これは直線ホモトピーと呼ばれる。図１５には輪郭線のホモトピー変形が示されている。この図において、一番上の輪郭線が形状関数ｆ、一方いちばん下の輪郭線はｇによって表されている。曲面はｆからｇへのホモトピーＦの軌跡として生成される。

【０１１８】（２）演算子をインプリメントするための要素

図１５の曲面を生成するための演算子は、ホモトピーによって輪郭線を変形するものとして記述することができる。

【０１１９】１．演算子を構成する要素

図１６には、演算子を構成する以下の主な４つの要素が描かれている。

【０１２０】（i）　ｆ：Ｉ→$R^3$　上の輪郭線の形状を与える

（ii）　ｇ：Ｉ→$R^3$　下の輪郭線の形状を与える

（iii）Ｆ：ｆからｇへのホモトピー

（iv）　ｈ：２つの輪郭線の高さの差

２．形状関数

ｆ、ｇとして以下の形状関数を準備する。

【０１２１】（i）　点　　：常に固定点の位置を与える定数関数

（ii）　円　　：円の形状を与える

（iii）多角形：任意の頂点を結ぶ多角形の形状を与える

（iv）　ベジェ：ｎ次元のベジェ曲線で、次式で記述される

【数３】

$$f(u)=\sum_{i=0}^{n}B_i^n(u)P_i \quad (0\le t\le 1)$$

この関数は制御点$P_i\in R^3$と呼ばれるｎ個の点の集合（順序つき）によって特定される。この制御点はユーザーによって修正することができる。ここで、$B^n{}_i$（t）はベルンシュタインの基底関数であり、次式で定義される。

【０１２２】

【数４】

$$B_i^n(u)=\binom{n}{i}u^i(1-u)^{n-i}$$

（v）ＮＵＲＢＳ（Non Uniform Rational B-Spline）曲線：この曲線の制御点もユーザーによって定義される。ＮＵＲＢＳ曲線は次の式で定義される。

【０１２３】

【数５】

$$f(u)=\frac{\sum_{i=0}^{n}N_{i,k}(u)w_iP_i}{\sum_{i=0}^{n}N_{i,k}(u)w_i}$$

ここで$W_i$は各制御点の重みである。$N_{i,k}$（t）はＢスプライン基底関数と呼ばれる（k-1）次の多項式の各区分の値を示す。これは次式で定義される。

【０１２４】

【数６】

$$N_{i,0}=\begin{cases}1 & (x_i\le u<x_{i+1})\\ 0 & otherwise\end{cases}$$

$$N_{i,k}(u)=\frac{(u-x_i)N_{i,k-1}(u)}{x_{i+k}-x_i}+\frac{(x_{i+k+1}-u)N_{i+1,k-1}(u)}{x_{i+k+1}-x_{i+1}}(2\le k\le n+1).$$

ＮＵＲＢＳは、非常に多くのＣＡＤシステムで用いられている。ＮＵＲＢＳは二次曲面を正確に表現することができ、また局所的な近似特性をもっている。すなわち、制御点またはそれに関連する重みが変化したとき、その点の近傍でしか曲面の形状に影響を与えない。

【０１２５】３．ホモトピーＦ

$$quadrant:\ F(x,t) = quadrant(x,t) = \sqrt{1-t^2}f(x) + (1-\sqrt{1-t^2})g(x)$$

（iii）放物線：$F(x,t)=(1-t^2)f(x)+t^2g(x)$

（iv）カージナルスプライン：カージナルスプラインは、一番上および一番下の輪郭線を内挿補間する。輪郭線間の対応点を示すトロイダルグラフを使うことによってパラメータの決定を自動化することができる。

【０１２７】（v）ガイディングカーブ：輪郭線上の点をガイディングカーブに沿って動かすことにより、輪郭線を変形することができる。輪郭線に対して複数のガイディングカーブを付けることができる。輪郭線がベジェ曲線またはＮＵＲＢＳ曲線で表されるとき、ガイディングカーブは制御点に付けられ、変形は制御点の動きによって決定される。ガイディングカーブが付けられていない制御点の動きは、隣接する制御点のガイディングカーブを用いて計算することができる。

【０１２８】図１７は、ユーザーがガイディングカーブを付けることにより、上の輪郭線が徐々に下の輪郭線に変形される様子を示している。

【０１２９】いずれの場合も、結果的に得られた輪郭線の間をカージナルスプラインを用いてパッチを当て、曲面を生成する。

【０１３０】４．演算子のための輪郭線の形状関数

演算子のための輪郭線の形状関数は以下のように与えられる。

【０１３１】（i）Ｐｕｔ_ｅ２

ｆ：点

ｇ：ユーザーが特定（デフォルト：円）

Ｆ：ユーザーが特定（デフォルト：四分円形）

（ii）Ｐｕｔ_ｅ０

ｆ：$e^0$が付けられる輪郭線の形状関数

$$c(t) = \begin{pmatrix} (1-\sqrt{1-(2-2t)^2})c(1/2)_x + \sqrt{1-(2-2t)^2}c(1)_x \\ (1-\sqrt{1-(2-2t)^2})c(1/2)_y + \sqrt{1-(2-2t)^2}c(1)_y \\ (2-2t)c(1/2)_z + (2t-1)c(1) \end{pmatrix}$$

で与えられる。この道の初期値のｘｙ平面への射影は、ｃ(0) および ｃ(1) を結ぶ線分になる。この ｃ(t) の式において、ルート（１／２乗）の部分を（$1-x^2$）に置き換えれば放物線の道を得ることもできる。Put_e1_divide および Put_e1_merge の要素は以下のとおりである。

【０１３２】・Put_e1_divide

ｆ：$e^1$が付けられる輪郭線の形状関数

ｃ：$e^1$

ｃ(0)、ｃ(1)：ｃ(0)＝ｆ（$s_1$），ｃ(1) ＝ｆ（$s_2$）なる $s_1$、$s_2 \in [0,1]$ で特定される

ｇ１、ｇ２：輪郭線を道ｃに沿って分割することにより得られる

・Put_e1_merge

ｃ：$e^1$の道

ｆ１、ｆ２：$e^1$が付けられる輪郭線の形状関数

ｃ(0)、ｃ(1)：ｃ(0)＝$f_1$（$s_1$），ｃ(1) ＝$f_2$（$s_2$）なる

$s_1$、$s_2 \in [0,1]$ で特定される

ｇ：道ｃに沿って輪郭線をマージすることにより得られる



一方、ホモトピーＦとして以下の関数が導入される。これらの関数は断面の輪郭線を出力する。

【０１２６】（i）線形：直線ホモトピー

（ii）四分円形：

【数７】

ｇ：点

Ｆ：ユーザーが指定（デフォルト：四分円形）

（iii）Put_e1_divide、Put_e1_merge

$e^1$の道ｃ：$[0,1]\rightarrow R^3$を決めなければならない。実際にインプリメントする場合、道ｃは ｃ(0)、ｃ(1/2) および ｃ(1) の位置によって特定される。道は滑らかでなければならず、また ｃ(1/2) における接線のベクトルはｘｙ平面に平行でなければならない。したがって、ｃ(1/2) は生成された曲面の鞍点になる。ｃ(1/2) の初期位置は、

【数８】

$$c(1/2) = \begin{pmatrix} \frac{c(0)_x+c(1)_x}{2} \\ \frac{c(0)_y+c(1)_y}{2} \\ c(0)_z - h \end{pmatrix}$$

である。ここで、

【数９】

$$c(t) = \begin{pmatrix} c(t)_x \\ c(t)_y \\ c(t)_z \end{pmatrix}$$

である。道の初期値は ｃ(0) と ｃ(1) を接続する楕円の弧であり、０≦ｔ≦１／２については、

【数１０】

$$c(t) = \begin{pmatrix} (1-\sqrt{1-(2t)^2})c(0)_x + \sqrt{1-(2t)^2}c(1/2)_x \\ (1-\sqrt{1-(2t)^2})c(0)_y + \sqrt{1-(2t)^2}c(1/2)_y \\ (1-2t)c(0)_z + 2tc(1/2)_z \end{pmatrix}$$

で与えられ、一方、１／２≦ｔ≦１については、

【数１１】

$$c(t) = \begin{pmatrix} (1-\sqrt{1-(2-2t)^2})c(1/2)_x + \sqrt{1-(2-2t)^2}c(1)_x \\ (1-\sqrt{1-(2-2t)^2})c(1/2)_y + \sqrt{1-(2-2t)^2}c(1)_y \\ (2-2t)c(1/2)_z + (2t-1)c(1) \end{pmatrix}$$

Put_el_divide による変形は、$e^1$の道をガイディングカーブとして用いることにより、輪郭線を変形することで行われる。すなわち、

・$F(s_1,t) = c(t/2)$

・$F(s_2,t) = c(1-t/2)$

である。一方、Put_el_merge の変形は次式によって得られる。

【０１３３】・$F_1(s_1,t) = c(t/2)$

・$F_2(s_2,t) = c(1-t/2)$

（iv）ダミーアイコン

すでに述べたダミーアイコンを用いることにより、輪郭線の構造を変えることなく輪郭線の形状を変形することができる。ダミー演算子の形状関数は以下のとおりである。

【０１３４】ｆ：ダミーが付けられる輪郭線の形状関数

ｇ：ユーザが指定（デフォルト：円）

このシステムは、子輪郭線が親輪郭線の境界を超えないことを自動的にチェックし、位相上の正しさを維持するものとする。

【０１３５】（３）微分不可能な場合および縮退への対応

これまで、物体表面の曲面が$C^2$級の可微分性をもつと仮定してきた。またすべての臨界点は非縮退であると仮定した。しかしながら、円筒や立方体など物体を設計する場合、多面体や、頂点または底辺が平面であるような微分不可能な点または縮退した点を含めて符号化できることが望ましい。

【０１３６】この観点から、縮退していない高さ関数をもつ$C^2$可微分の多様体をもとに機能の拡張を行う。具体的には以下のとおりである。

【０１３７】◎ 微分不可能な点を含む輪郭線

図１８は微分不可能な点を含む物体を示す。この場合、図１９に示すような微分可能な形状関数をもつ物体に置き換えることによって符号化が可能となる。

【０１３８】◎ 水平な頂部、底部または分岐部

水平面を表す場合には、高さｈの変化を０に置き換えればよい。例えば図２０に示すように、頂部も分岐部もそれぞれ水平面の場合、図２１のように高さを０として表現することができる。

【０１３９】◎ 尾根線（リッジ）

図２２に示すようにリッジに対応する場合、形状関数ｆおよびｇを、途中で折り返す線分$\alpha : I \rightarrow R^3$を用いて、以下のように設定することができる。すなわち、１本のひもを２つ折り畳んだような形状である。

【０１４０】$f(x), g(x) = \alpha(2x)$　　$0 \leqq x \leqq 1/2$

$f(x), g(x) = \alpha(1-2x)$　　$1/2 < x \leqq 1$

◎ 火山のリム

図２３に示すように火山のリムについては、リッジ同様の方法および高さｈを０にすることによって記述が可能となる。



【０１４１】［３］断面データからの滑らかな曲面の生成

（１）はじめに

断面データからもとのオブジェクトを再構成したい場合がある。特に医療分野では、いくつかの断面データからもとの器官の全体形状を再生する意義が大きい。３Ｄ医療データの生成方法として次のサーフェスモデルが知られている。

【０１４２】１．三角形タイル手法

多角形で表現された隣接輪郭線どうしの間を三角形パッチで埋めていく。

２．スプライン近似

スプライン近似によって曲面を構成する。

ここではホモトピー、および特別な場合としてこれら２つのサーフェスモデルを包含する新たなモデルを提案する。このモデルは、これら２つのサーフェスモデルの欠点を解消するものである。

【０１４３】このホモトピーモデルは、連続トロイダルグラフによる輪郭線上の対応点対の表示と、その表示からホモトピーを用いて曲面を生成するステップを含む。最終的に得られる曲面は古典的なパラメトリックな曲面であり、スプライン近似を包含している。ここでは２つの例を示す。１つは直線ホモトピーを用いてロフト曲面を洗練するもの、もう１つはカージナルスプライン曲面である。

【０１４４】（２）トロイダルグラフによる表現

三角形手法については、トロイダルグラフによる表現が知られている。このグラフ理論では最短対角線アルゴリズムに基づく手法が提案されている。まず始めに、既知のトロイダルグラフ（これは連続グラフではない）について説明し、つぎに我々が拡張した連続バージョンを説明する。

【０１４５】ここで、輪郭線はひとつづきの線分によって近似されるものとする。また、ある輪郭線をｍ個の異なる輪郭線点$P_0$、$P_1$、…$P_{m-1}$の列によって定義し、別の輪郭線を$Q_0$、$Q_1$、…$Q_{n-1}$によって定義する。これら２つの閉曲線の方向は同じであると考える。三角形の生成は次の２つの条件を満たさなければならない。

【０１４６】１．同一輪郭線上の２つのノードが同じ三角形のノードとして定義されるためには、これらのノードがその輪郭線上で隣合っていなければならない。

【０１４７】２．どの三角形についても、頂点のうちの２つだけが同じ輪郭線から取得される。

【０１４８】従来のトロイダルグラフでは、これら２つの規則を統合した上でグラフ理論に反映されている。すなわち、三角形相互の位相上の関係をトロイダルグラフの中に表現するのである。このグラフでは、頂点は$P_0$、$P_1$、…$P_{m-1}$と$Q_0$、$Q_1$、…$Q_{n-1}$の間の可能なスパンすべての組に対応し、弧は可能な三角形すべての組に対応する。例えば、頂点（$P_3$，$Q_5$）は$P_3$と$Q_5$を結

ぶスパンを示し、頂点（$P_3$，$Q_5$）から（$P_3$，$Q_6$）の弧によって△$P_3Q_5Q_6$の三角形ができる。

【０１４９】曲面として許容可能なもののグラフでは、すべての行において垂直方向の弧がちょうど１つ存在し、またすべての列において水平方向の弧がちょうど１つ存在する。輪郭線間に生成される曲面のうち、許容できるものは２とおりである。１つは円筒と位相同形のもの、もう１つは２つの円錐と位相同形のものである。以下、前者の場合を中心に説明する。

【０１５０】最短対角線アルゴリズムは単純では、まず隣接する輪郭線がそれぞれ単位正方形に写像される。ここで点$P_0$と、$Q_0$が近いと仮定する。最短対角線アルゴリズムでは、点$P_0$と点$Q_0$を結ぶことによって三角形の生成を開始する。点$P_i$および$Q_j$が接続された後、つぎに接続すべき候補である$P_i-Q_{j+1}$、または$P_{i+1}-Q_j$の一方が選択され、接続される。以降、この繰り返しである。これを平面上で考えれば、階段関数によって表現できる。ただし、輪郭線を一周すればもとに戻るため、階段の開始点と終了点は同一点でなければならない。このことを表現するために、Ｐをトーラスの外周を円周に沿って移動する方向、Ｑをそれと垂直方向にとることができる。この表現方法の場合、Ｐがトーラスの外周を一周したとき、Ｑもトーラスの曲がった円筒部分を一周し、もとの点に戻る。

【０１５１】以上、この方法は、隣接する閉曲線どうしの形状が近い場合には極めて良好な結果をもたらすが、それらの形状が大きく変化する場合には、無理な対応づ

・定義１：$d(P_i, Q_j)=\min d(P_i, Q_k)$　$(0\leqq k<n)$

$d(P_i, Q_j)=\min d(P_k, Q_j)$　$(0\leqq k<m)$

がともに成り立つとき、頂点（$P_i$，$Q_j$）を最近点対とよぶ。ここで、ｄ（Ｐ、Ｑ）は点Ｐと点Ｑの間の距離である。

【０１５６】・注意１：トロイダルグラフの各行および各列に存在する最近点対は、せいぜい１つである。

【０１５７】一般に、すべての最近点対を結ぶ許容可能な経路は存在しない。そのためできる限り多くの最近点対を結ぶ方法が検討される。そのためまず、最近点対である点の重みを１、それ以外を０とする。Ψ（Ｓ）をＳが通過する頂点の重みの合計と定義する。いいかえれば、Ψ（Ｓ）は許容可能な経路Ｓが通過する最近点対の数である。この結果、問題はコスト許容性が最大になる経路を見つけることに還元される。

【０１５８】もちろん理論的にはそれで正しいが、そのための計算量は非常に大きい。そこで我々は、最近点対をグループ化する、より単純なアプローチを考案した。実験の結果、最近点対がグループ化できることが判った。そのグループとは、$P_i$と$Q_j$以降、それぞれ次の点どうしが最近点対となる（$P_i$，$Q_j$）、（$P_{i+1}$，$Q_{j+1}$）、…（$P_{i+k}$，$Q_{j+k}$）から構成されるようなグループである。これを「長さＬ（Ｒ）がｋ＋１である最けが原因となり、生成される曲面に人工的な皺、またはフォールド（折り目）が生じることが知られている。

【０１５２】この様子は図２４の$q_3-p_0-q_2$に示される。同図のごとく、曲面は点Ａの回りにしわを持つように見える。人工的な皺はスムーズシェーディングを用いても消すことができない。この問題はノードの接続状態を修正することで解決することはできない。なぜなら、$p_0$および$p_1$の間にノードが存在しないためである。

【０１５３】このように、三角形手法によって生成される曲面は各輪郭線に対するノードの選択に大きく依存する。三角形手法は、輪郭線を局所的に見ながら三角形をつないでいくため、上記の皺のような局所的問題が生じる。なお、各輪郭線の外周がちょうど１になるように輪郭線を正規化することによって、輪郭線間のコヒーレンス（連接性）を改善する提案もある。しかしその方法でも輪郭線上の局所的な制約条件のみが考慮されている。三角形手法では、生成される曲面の滑かさも問題になる。スムーズシェーディングを施された三角形表現の曲面は一見滑らかに見えるが、曲面の法線は曖昧になり、オブジェクトの形状が複雑な場合、それを正確に把握することが困難になる。

【０１５４】（３）最適経路を見出すための発見的アプローチ

以下、経路とは対応しあう輪郭線上の対応しあう点を結ぶ線をいう。ここではまず、「最近点対」を定義する。

【０１５５】

近点対のひとづづきのランＲ」と表現し、点ｐがランＲの構成要素であるときｐ∈Ｒと表記する。このとき次の命題が成り立つ。

【０１５９】命題１．最近点対（$P_i$，$Q_j$）を通過する許容可能な経路$S_1$が存在するとき、ランＲのすべての最近点対を通過する許容可能な別の経路$S_2$が存在する。ただし、（$P_i$，$Q_j$）∈Ｒであり、Ψ（$S_1$）≦Ψ（$S_2$）である。

【０１６０】[証明]Ｌ（Ｒ）≧２であるとして、一般性を失わない。

【０１６１】ケース１：（$P_{i+1}$，$Q_{j+1}$）∈Ｒ、かつ$S_1$がこの点を通らない場合

ケース１ａ：$S_1$が（$P_{i+1}$，$Q_j$）を通る場合

$S_1$が列ｊ＋１を離れるとき、最後に通る点を（$P_k$，$Q_{j+1}$）とする。また、$S_1=S_3S_4S_5$とする。ここで$S_4$を（$P_{i+1}$，$Q_j$）～（$P_k$，$Q_{j+1}$）の経路とすれば、$S_4$は（$P_i$，$Q_j$）も（$P_{i+1}$，$Q_{j+1}$）も通らないため、注意１によりΨ（$S_4$）＝０となる。一方、（$P_{i+1}$，$Q_{j+1}$）∈Ｒであるため、明らかに（$P_{i+1}$，$Q_{j+1}$）～（$P_k$，$Q_{j+1}$）の経路であってΨ（$S_6$）＝１となる$S_6$が存在する。それゆえ、$S_2$を$S_3$、$S_6$、$S_5$を含む（す

なわち、$S_4$のかわりに$S_6$を含む）経路とすれば、Ψ（$S_2$）＝Ψ（$S_1$）＋１である。

【０１６２】ケース１ｂ：$S_1$が（$P_1$，$Q_{j+1}$）を通る場合

ケース１ａと同じである。

【０１６３】ケース２：（$P_{i-1}$，$Q_{j-1}$）∈R、かつ$S_1$がこの点を通らない場合

ケース１と同様である。

【０１６４】この構成方法を再帰的に適用することにより、命題１の経路$S_2$が得られる。命題１によれば、Ψを最大にするためにランのみを考慮すればよく、各ランの中身までは考えなくてよいことが判る。したがって前に述べた目的、すなわち、できる限り多くの最近点対を結ぶ方法の検討は、Ψを最大にする許容可能なランの集合を選択することに還元される。

【０１６５】ここで、許容可能な経路が通過するランの集合を許容可能なランの集合と定義する。明らかに、２つのランから構成されるランの集合はいずれも許容可能である。３つのラン｛$R_1$，$R_2$，$R_3$｝の集合であって許容可能でないものは以下のとおりである。まず、$R_1$＝（$P_{i1}$，$Q_{j1}$）…、$R_2$＝（$P_{i2}$，$Q_{j2}$）…、$R_3$＝（$P_{i3}$，$Q_{j3}$）…、および$i_1 < i_2 < i_3$（mod m）と仮定する。ここで、$j_1 < j_3 < j_2$（mod m）または$j_2 < j_1 < j_3$（mod m）または$j_3 < j_2 < j_1$（mod m）であるとき、この集合は許容可能ではない。mの剰余系を考えるのは、輪郭線が一周するともとに戻るためである。

【０１６６】４以上のランで構成される集合については、それに含まれるいかなる３つの構成要素も許容可能な集合である場合に限り、許容可能である。このことからわかるように、この計算労力は大きい。

【０１６７】こうした複雑性に鑑み、我々は発見的なスキームを考えることにした。複雑さの原因の１つはmに関する剰余を用いる点にあるため、これを廃止する。これはトーラス上にP，Qを表したとき、そのトーラスを切断して平面にすることに対応する。この単純化により、許容可能性はランの三値的な問題ではなく、二値的な問題になる。２つのラン（$P_{i1}$，$Q_{j1}$）…、および（$P_{i2}$，$Q_{j2}$）…の集合は、$i_1 < i_2$かつ$j_1 < j_2$、または$i_1 > j_2$かつ$j_1 > j_2$の場合、許容可能である。集合｛$R_1$，$R_2$｝が許容可能でないとき、$R_1$は$R_2$と不適合であると表現する。このテスト方法は極めて簡単であるが、トーラスを２つに切断して平面にするとき、ランが分割される可能性がある。そこで最終的に、Ψを最大にするランを選択するために、スタックを用いた以下の発見的アルゴリズムを採用する。

【０１６８】<u>アルゴリズム</u>

N個のランがあり、各ラン$R_k$＝（$P_{ik}$，$Q_{jk}$）について$i_1 < i_2 < \cdots < i_k < \cdots < i_n$が成り立つとする。また最初にスタックは空であるとする。

【０１６９】ステップ１．$R_1$をスタックに積む。



ステップ２．ｋを２とする。

ステップ３．Ｒをスタックの一番上に積まれたランであるとする。

ステップ４．もし$R_k$とＲが適合すれば、$R_k$をスタックに積む。

そうでない場合であって、かつＬ（$R_k$）≦Ｌ（Ｒ）であれば、$R_k$がスタックから取り除かれ、ｋ＝ｋ＋１とする。それ以外の場合、Ｒがスタックから取り除かれる。

【０１７０】ステップ５．ｋ≦Ｎであればステップ３に戻る。

【０１７１】$R_k$については、ｋ回以下の繰り返しが必要である。そのため、このアルゴリズムの時間複雑性はＯ（$N^2$）である。このアルゴリズムにより、互いに適合するランのうち、ある長さを超えるものが次々にスタックに積まれ、確保されていく。この結果、Ψをほぼ最大値にする許容可能なランの集合を得ることができる。最終的に得られたランの様子は図２５に示される。

【０１７２】◎ トロイダルグラフの連続バージョン

つぎに、経路の進路を詳細に決める必要がある。経路は上述のアルゴリズムによって選ばれたランの各点を通過しなければならないが、それ以外の点についてはまだ未定の状態にある。ここでは進路を決めるために、最近点対どうしの間の線形補間を採用する。詳細な説明に入る前に、離散的なトロイダルグラフの連続バージョンの概要を説明する。

【０１７３】まず始めに、上下の輪郭線はなんらかのパラメータによって表示されなければならない。輪郭線上の点は以下の関数によって指定される。

【０１７４】ｆ，ｇ：Ｉ→$R^3$、ここでＩ∈Ｒ、かつ［０，１］で定義される。また ｆ(0)＝ｆ(1)、およびｇ(0)＝ｇ(1)である。

【０１７５】連続トロイダルグラフでは、そのグラフに含まれる２つの点の水平および垂直距離がそれら２つの間のパラメータ値の差を表す。したがって、パラメータがＮＵＲＢＳ曲線などのパラメータそのものを指す［２］とは意味合いが異なる。パラメータの例は以下のとおりである。

【０１７６】・例１：弧長をパラメータとして使う。$l_1$と$l_2$がそれぞれ下と上の閉曲線の長さを表すとする。例えば、（$P_{i1}$，$Q_{j1}$）と（$P_{i2}$，$Q_{j2}$）のグラフ上の水平距離は上の閉ループ上の$P_{i1}$～$P_{i2}$の弧長を表す。またグラフ上の垂直距離は下の閉ループ上の$Q_{j1}$～$Q_{j2}$の弧長を表す。グラフ上の点（ｘ，ｙ）は$P_0$からの弧上の距離がｘ$l_1$であるような点Ｐ、および$Q_0$からの弧上の距離がｙ$l_2$であるような点Ｑの組を表すよう正規化される。

【０１７７】・例２：輪郭線の形状が円に近い場合、偏角をパラメータに用いることができる。この場合、本質的には円筒座標系を用いることと同じである。なお、輪

郭線は線分で近似する必要はない。パラメトリックな曲線、例えばベジェ曲線、Ｂスプライン曲線またはカージナル・スプライン曲線を利用することができる。

【０１７８】・例３：スプライン基数関数のパラメータをグラフのパラメータとして利用する。

【０１７９】後述する線形補間を利用するためには、弧長による表示が望ましい。しかし、曲線パラメータから弧長への変換は容易ではない。１つの解決策は、スプライン曲線を線分によって近似し、実際の弧長の代りに線分の長さを利用する方法である。後に示す表示例はこの近似を用いている。

【０１８０】点が輪郭線上を最初のノードから離れるに従ってパラメータが単調増加する限り、以下の議論においてどのようなパラメータを使うかは本質的な問題ではない。許容可能な経路に関するグラフの連続バージョンは、このグラフ上で単調増加（または単調減少）する「多値関数」として表現される。厳密にいえば、経路はグラフ$y=U_i(x)$ および$x=V_i(y)$ の連鎖として表示される。ここで、

$U_i$：$[x_{2i},\ x_{2i+1}]\rightarrow I$

$V_i$：$[y_{2i},\ y_{2i+1}]\rightarrow I$

（$0=x_0\leqq x_1\leqq\cdots\leqq x_{2nn}-1<1$，$0=y_0\leqq y_1\leqq\cdots\leqq y_{2nn-1}<1$）は、単調増加（単調減少）関数であり、$U_i$および$V_i$の結合点（$x_{2i+1}$，$y_{2i+1}$）において、

$U_i(x_{2i+1})=V_i(y_{2i+1})$、および

$U_0(0)=V_{nn-1}$ (1)

が成立し、$V_i$および$U_{i+1}$ の結合点（$x_{2i+2}$，$y_{2i+2}$）において、

$V_i(y_{2i+2})=U_{i+1}(x_{2i+2})$

が成立するような関数である。

【０１８１】図２６はｎｎ＝２の場合を示す図である。なお、離散的なトロイダルグラフはこの連続バージョンの特別な場合であり、変換は階段状、すなわち$y=q_i$ および$x=p_i$ の連鎖という形で表される。

【０１８２】以上の考察をもとに、経路の詳細が最終的にこのグラフ上で決定される。最近点対が（$x_i$，$y_i$）（ｉ＝０，１，…ｋ－１）であり、$0=x_0<\cdots<x_{k-1}$であると仮定する。この経路は以下のような関数$U(x)$：$I\rightarrow I$によって記述することができる。

【０１８３】

【数１２】

$$U(x)=(y_{i+1}-y_i)\frac{(x-x_i)}{(x_{i+1}-x_i)}+y_i(x_i\leq x<x_{i+1}).$$

すでに述べたとおり、この式は（$x_i$，$y_i$）と（$x_{i+1}$，$y_{i+1}$）の間の線形補間である。この様子は図２７に示される。この経路によって表現される表面は図２８に示される。輪郭線自身は図２４と同じである。この例は直線ホモトピーを表している。図２４と図２８を比較すれば明らかなとおり、図２８の表面の方が図２４に比べてより滑かである。「皺」の発生も回避されている。



【０１８４】なお、定義１の最近点対は離散的な点どうしの距離の比較から決定したが、この概念も連続トロイダルグラフに合わせて拡張することができる。すなわち、最近点対を「輪郭線上で互いに最も近い点の対」と定義すればよい。ただし、最近点対は無数に存在しうるため、実際のインプリメンテーションでは適度なサンプリングを行うものとする。

【０１８５】以上、我々の手法では、まず最近点対間の対応をとり、つづいてそれ以外の点の対応をとるというステップを経る。このため、局所的な対応状態に注目する従来の手法で問題となる不自然な接続結果が生じることはい。また、最初に輪郭線全周に渡って最近点間の対応をとるため、輪郭線の局所構造に起因するノイズに対する耐性が向上する。

【０１８６】（４）ホモトピーモデルに基づく表面の生成

すでに述べたとおり、連続トロイダルグラフ上の点は上下の輪郭線の点の間の対応を示す。対応点どうしはホモトピーによってつながれる。正確にいえば、曲面はｆ～ｇＵへのホモトピーの軌跡によって生成される。ここでＵは連続トロイダルグラフ上の関数であり、輪郭線どうしの対応を示す。直線ホモトピーの場合、それらの点は単純に直線によって接続される。以下、ホモトピーモデルを用いた表面生成を説明する。

【０１８７】このホモトピーモデルでは、ホモトピーを用いて各輪郭線上の対応点どうしを接続することにより、隣接する輪郭線の間に表面パッチを生成する。このモデルでは離散的なトロイダルグラフによる対応点対の表現とは異なり、輪郭線上のすべての点が隣接する輪郭線上に対応点を持つため、ホモトピーがそれらの対応点間を接続するファイバーとして利用される。輪郭線間の表面パッチはホモトピーの軌跡と見なすことができる。わかりやすい例は円筒である。円筒の表面は２つの円の間の対応点を接続することで生成できる。詳細は省くが、このモデルはパラメトリックな曲面を包含している。

【０１８８】◎ ホモトピーを用いたパラメトリックな曲面の生成

下および上の輪郭線を写像ｆ，ｇ：$X\rightarrow R^3$、$X=[x_0,\ x_1]\in R$で表す。

【０１８９】簡単のために、下の輪郭線はｚ＝０の平面上、上の輪郭線はｚ＝１の平面上にあるとし、Ｘ＝Ｉとする。「２つの点Ｐ（$p_x$，$p_y$）、Ｑ（$q_x$，$q_y$）がホモトピーＦによってつながれる」とは、

$F(x,0)=f(x)=(p_x,\ p_y,\ 0)$

$F(x,1)=g(x)=(q_x,\ q_y,\ 1)$

がともに成立するような$x\in I$が存在することである。すなわち、上下の輪郭線の間にあり、平面ｚ＝ｔ上のす

べての断面がｘ∈Ｘなる Ｆ(x,t) によって与えられるとき、Ｆはｆからｇへのホモトピーである。いいかえれば、表面は２変数関数 Ｆ(x,t) によって表される。

【０１９０】２つの輪郭線がともに１個の閉曲線からなる場合、それらは位相同形である。一方、２つの輪郭線が異なる数の閉曲線を含む場合、それらは位相同形ではない。そのような場合、分岐に対する取扱いが必要になる。簡単のために、上の輪郭線は２つの閉曲線、下の輪郭線は１つの閉曲線を含むと仮定する。分岐に対する取扱いは、２つの閉曲線が１点で結合するような仮想輪郭線を上下輪郭線の間に導入することによって可能となる。図２９に示すとおり、この結合箇所が１点である場合、仮想輪郭線は２つの閉曲線を含む。またこの結合箇所が２つの別個の点と考えられる場合、仮想輪郭線は１つの閉曲線からなると考えてよい。

【０１９１】◎ トロイダルグラフ表示からの表面の生成

ｆ，ｇ：Ｉ→$R^3$が下および上の輪郭線を表す関数とする。ホモトピーモデルの場合、考慮すべき許容可能な経路は逆関数$U^{-1}$を持つ単調増加連続関数Ｕ：Ｉ→Ｉによってトロイダルグラフ上に表現されなければならない。

$$(w_{-1}(t)w_0(t)w_1(t)w_2(t)) = (t^3t^2t1)\begin{pmatrix} -a & 2-a & -2+a & a \\ 2a & -3+a & 3-2a & -a \\ -a & 0 & a & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

である。$f_1$がカージナルスプライン関数によって表現されるとき、その表面をカージナルスプライン表面と呼ぶ。

【０１９４】<u>［４］断面からのレーブグラフの自動的な生成</u>

（１）はじめに

この章では、オブジェクトの断面データが与えられたときに、そのオブジェクトの符号を自動的に取得する方法を説明する。この技術により、断面データが利用できるとき、自然界に存在するオブジェクトの形状を取扱うことができる。まず最初に、オブジェクトのレーブグラフを自動的に再構成する。レーブグラフが構成されれば、そこから表面の符号を得ることは容易である。ここでも$C^2$連続性と非縮退の特異点を仮定する。

【０１９５】一連の処理は［２］で説明した符号化データから表面を生成する処理の逆演算と見なすことができる。このためユーザーは、表面を直接符号化する代りにレーブグラフまたは断面を記述することができる。この方法はまた、得られた結果が、直接断面データを用いた表面再構成システムに与えられる場合にも有益である。

【０１９６】医療分野への応用を考えたとき、表面再構成システムに対してデータを準備する工程は以下のとおりである。すなわち、ＣＴスキャンで器官の画像を撮影して登録する。つぎに輪郭線の外形線をデジタイズし、いずれの輪郭線どうしを接続するかを決定する。この



しかる後、ｆ(x)および ｇ(Ｕ(x)) の間のホモトピーによって表面が生成される。直線ホモトピーが使用されるとき、これは次の式で表現される。

【０１９２】

【数１３】

$$\bar{F}(x,t) = \begin{cases} (1-t)f(x) + tg(0) & x \le t \\ (1-t)f(0) + tg(x) & x > t \end{cases}$$

カージナルスプラインを用いることもできる。４つの輪郭線を$f_{-1}$、$f_0$、$f_1$、$f_2$で表し、それぞれの間の許容可能な経路を$U_{-1}$、$U_0$、$U_1$でそれぞれ表すとする。この場合$f_0$と$f_1$の間の表面は次の式で表現される。

【０１９３】

【数１４】

$$\begin{aligned} F(x,t) &= w_{-1}(t)f_{-1}(U_{-1}^{-1}(x)) \\ &+ w_0(t)f_0(x) \\ &+ w_1(t)f_1(U_0(x)) \\ &+ w_2(t)f_2(U_1(U_0(x))) \end{aligned}$$

ただし、

【数１５】

後、スムージングによって登録されていないデータを計算し、最後に表面を構成する。

【０１９７】この一連のプロセスの中で最も難しい部分は、いずれの輪郭線対に表面パッチを生成すべきかを決定することである。この目的のためにレーブグラフが利用される。レーブグラフでは各輪郭線がノードとして表現され、２つの連続する断面の輪郭線間の位相上の関係を示すことができる。

【０１９８】既述の図４（ａ）～（ｃ）は、例としてトーラスの高さ関数のレーブグラフを示す。この例では、レーブグラフの点は断面上の輪郭線を表している。例えば人の内耳の半規管のように、データが複雑な場合、このレーブグラフを人手作業で生成することは大変な労力を要するとともに、非常に難しい作業になる。圧倒的な枚数の断面画像からオブジェクトの全体的な位相構造を理解することはほとんど不可能に近い。

【０１９９】位相構造の理解を容易ならしめるために、我々はレーブグラフの自動再構成方法を開発した。隣接する断面上に存在する任意の輪郭線対を考えるとき、この対の重み関数を定義する。我々の方法は、この重み関数および表面にあいたハンドル（つまり穴）の数に関する先験的な知識を用いることによって、レーブグラフを自動的に構築することができる。まず始めに、我々のアルゴリズムは輪郭線の数が変化しないような、レーブグラフの辺の主要部分を生成する。その後、上述の重み関

数およびハンドルに関する先験的な知識を用いてグラフの残りの部分を決定する。具体的には、既知のハンドルの数に矛盾しないような辺を、重みが減っていく順に（すなわち、最初に重みが最大の箇所に辺を付けるように）、グラフに付け加えることによってレーブグラフを完成させる。

【０２００】（２）レーブグラフの定義

レーブグラフは立体オブジェクトの位相上の「骨格」を表し、いずれの輪郭線の間に表面パッチを生成すべきかを示す。

【０２０１】レーブグラフは多様体の上に定義される。多様体は、ユークリット空間と同じ局所的特性を持つとみなすことができる。例えば、平面や球面は二次元多様体である。以下、立体オブジェクトの表面は二次元多様体であると仮定する。

【０２０２】ここでは、オブジェクトの二次元多様体の表面上の高さ関数ｈ（Ｘ）に関するレーブグラフを用いる。ここでｈ（Ｘ）はオブジェクトの表面上の点の高さを与える関数で、Ｘ＝（ｘ，ｙ，ｚ）はＲ$^3$（ｘ，ｙ，ｚ∈Ｒ）に埋め込まれている。すなわち、ｈ（ｘ，ｙ，ｚ）＝ｚである。

【０２０３】表面上の高さ関数のレーブグラフは、Ｒ$^3$におけるグラフの商空間である。商空間は２つの点（ｘ$_1$，ｙ$_1$，ｚ）および（ｘ$_2$，ｙ$_2$，ｚ）が高さｚにおいて曲面の断面上の同じ連結成分であるとき、これらを等化する（図４（ａ）～（ｃ））。これは、図４（ｂ）に示すようにｚ軸に対して垂直な断面を考えればわかりやすい。すなわちレーブグラフは、各断面における輪郭線をノードとして表現する。オブジェクトの表面に開いている穴（ハンドル）はすべての臨界点が非縮退であり、かつ各鞍点に集まる辺の数が３であるとき、レーブグラフにおいて閉曲線として現れる。この知見を後述のアルゴリズムに利用する。

【０２０４】レーブグラフは、いずれの輪郭線の間に表面パッチが存在するかを示している。以下、底部からｉ番目の断面を含む平面の式をｚ＝ｚ$_i$と仮定し、この平面を第ｉフレームと呼ぶことにする。

【０２０５】（３）レーブグラフの構成

◎ 重み関数

ここでは一連の断面が与えられたときにレーブグラフを構成する重み関数を定義する。隣接する断面に含まれる輪郭線の対が与えられたとき、この関数は重みを与える。分岐を取扱うとき、重みが減っていく順に輪郭線の間に辺を張っていく。

【０２０６】重み関数の作り方には何とおりかの方法がある。重み関数は、互いにより近い輪郭線の対に対してより大きな重みを与えるものとする。おおまかにいえば、我々は輪郭線間の「平均距離」の逆数を用いた。以下の計算において、我々はこの平均距離を輪郭線間の距離Ｄによって割った。このことにより、異なる断面の輪



郭線の対の重みを比較することができる。

【０２０７】第ｊフレームの第ｉ輪郭線を$L^j_i$で表現する。また、$L^j_i$および$L^{j+1}_k$の間に生成される表面パッチに対応する辺の重さをｑ（$L^j_i$，$L^{j+1}_k$）で表す。$N^j$は第ｊフレームに含まれる輪郭線の数、$N_{frame}$はフレームの数を表す。輪郭線は多角形によって近似されるものとし、実際のインプリメンテーションでは以下の式１を用いた。

【０２０８】

【数１６】

$$q(L_i^j, L_k^{j+1}) = n_i^j D_j / \sum_{m=0}^{n_i^j} f(x_{i,m}^j, L_k^{j+1}).$$

ここでＤ$_j$は第ｊフレームと第（ｊ+1）フレームの間の距離である。$L^j_i$を近似する多角形は$n^j_i$個の頂点を持ち、第ｍ点を$x^j_{i,m}$で表す。ｆは次式で定まる。

【０２０９】

【数１７】

$$f(x, A) = \min_{y \in A} d(x, y)$$

この式においてｄは距離の関数、ＡはＲ$^3$に含まれる点の集合、ｘ，ｙ∈Ｒ$^3$とする。この関数ｆはｘと集合Ａの中の点の最短距離を与える。

【０２１０】◎ レーブグラフの自動生成

以下、重み関数と任意の一連の断面を用いて我々のアルゴリズムによるレーブグラフの構成を説明する。始めに、つぎに示すケース１のごとく、輪郭線の数が変わらない主要な部分についてグラフを生成する。数が変わる場合は、ケース２で対応する。すなわち、予めわかっている穴の数と矛盾しない辺を、重みの減る順番にレーブグラフの中に張っていく。

【０２１１】ケース１：$N^j = N^{j+1}$の場合

ケース１ａ：輪郭線が互いに最も近い$L^j_{i1}$、$L^{j+1}_{i2}$の対で構成される場合。

すなわち、

【数１８】

$$q(L_{i1}^j, L_{i2}^{j+1}) = \max_{k=0}^{k=N^{j+1}-1} q(L_{i1}^j, L_k^{j+1})$$

$$q(L_{i2}^{j+1}, L_{i1}^j) = \max_{k=0}^{k=N^j-1} q(L_{i2}^{j+1}, L_k^j)$$

がともに成り立つ場合（このような対を「最近輪郭線対」と呼ぶ）。

【０２１２】重み関数を用いることにより、一対の輪郭線の間が辺で結ばれる。図３０（ａ）～（ｅ）はケース１、２を説明する図である。ここでは、図３０（ａ）および（ｂ）のような状況は発生しないと仮定する。

【０２１３】ケース１ｂ：ケース１ａが成り立たない場合

図３０（ｃ）に示すような分岐が生じていると考える。

我々のインプリメントテーションでは、次の条件のいずれかが成立する場合、$L^{j}_{i1}$ および $L^{j+1}_{i2}$ の間に辺が張られる。

【０２１４】

【数１９】

$$q(L^{j}_{i1}, L^{j+1}_{i2}) = \max_{k=0}^{k=N^{j+1}-1} q(L^{j}_{i1}, L^{j+1}_{k})$$

$$q(L^{j+1}_{i2}, L^{j}_{i1}) = \max_{k=0}^{k=N^{j}-1} q(L^{j+1}_{i2}, L^{j}_{k})$$

ケース２：$N^{j} \neq N^{j+1}$ の場合

ケース２ａ：２つのフレームの間に分岐が起っている場合（図３０（ｄ））

ケース２ｂ：分岐が起っていない場合（図３０（ｅ））

これら２つのケースは重み関数によって区別することができる。ｑ（$L^{j}_{i1}$，$L^{j+1}_{k}$）、ｑ（$L^{j}_{i2}$，$L^{j+1}_{k}$）がともに大きな数値の場合、$L^{j+1}_{k}$ から $L^{j}_{i1}$、$L^{j}_{i2}$ への分岐が起っている可能性が高い。

【０２１５】このように重み関数のみからでも分岐の可能性を判断することは可能である。しかし我々は、重み以外にオブジェクトの大域的な情報を利用するアルゴリズムを提案する。以下、$N_{conn}$ はオブジェクトの連結成分の数、ｇはオブジェクトの穴（ハンドル）の数、$M_{conn}$ はレーブグラフの連結成分の数、１はレーブグラフの閉曲線の数を示す。つぎに示す９ステップのアルゴリズムでは、$N_{conn}$ ＝１であると仮定する。また最初、グラフには辺がない状態であるとする。

【０２１６】アルゴリズム

・ステップ１：隣接するフレーム上の輪郭線のすべての対について関数ｑを計算し、すべての輪郭線 $L^{f}_{c}$（ｆはフレーム番号、ｃは輪郭線番号）についてその最も近くに隣接する輪郭線 $L^{f'}_{c'}$（ｆ´はフレーム番号＋ｄｉ、ｃ´は最近輪郭線番号）を示す表を作成する。$L^{f'}_{c'}$ はｑの最大値およびｄｉ＝±１におけるｑの値を与える。

【０２１７】・ステップ２：まずケース１のすべての対に関して辺をグラフに追加する。

【０２１８】・ステップ３：つぎにケース２にあたる輪郭線 $L^{j}_{i1}$ および $L^{j+1}_{i2}$ の対を選択し、最近輪郭線対の間に辺を張る。

【０２１９】・ステップ４：１＞ｇまたは $M_{conn}$ ＜$N_{conn}$ の場合、アルゴリズムはステップ９に飛ぶ。

【０２２０】・ステップ５：ステップ３において辺が張られなかったケース２の輪郭線の対のリストを重みが大きい順にならべて生成する。こうして表示される辺を $E_1$、$E_2$…と書く。

【０２２１】・ステップ６：作成されたリストの中から一番上にある輪郭線の対を選択する。

【０２２２】・ステップ７：その対に対して辺を張ったとき、$1 \leqq g$ かつ $M_{conn} \geqq N_{conn}$ なら、その辺をグラフに追加する。そうでない場合、その辺を拒絶する。

【０２２３】・ステップ８：処理が終了した対をリストから削除する。リストが空にならなければ、ステップ６に戻る。

【０２２４】・ステップ９：生成されたグラフが期待どおりでない場合、接続されるべきでない輪郭線の対の重みを対話形式で０に修正する。しかる後、処理をステップ１に戻す。

【０２２５】以上のアルゴリズムでレーブグラフが構成できれば対応あう輪郭線が判明するため、後は［３］の表面構成技術によってオブジェクトを再構成することができる。なお、このアルゴリズムを $N_{conn}$ ＞１である複数要素のオブジェクトに適用する場合、ステップ４において各連結成分に対してｇを指定すればよい。

【０２２６】◎ 医療分野への適用

我々はこのアルゴリズムを用いて、人の耳の蝸牛管および半規管の三次元表面を再構成した。セロイジン（切片固定剤）処理により、耳の標本を固定した上で、これを２０μｍｍの厚さにスライスした。この標本を写真撮影し、グラフィカ社のドラムスキャナ（Ｇ－２２５Ｃ）を用いて写真をイメージデータに変換し、スタイラスペンを使ってオブジェクトの外形線をプロットした。処理には、ＨＰ９０００シリーズのモデル５５０を用いた。最後に輪郭線データをシリコングラフィックス社のパーソナルアイリスに入力し、レーブグラフを生成してオブジェクトの表面を再構成した。

【０２２７】図３１は断面データを基に構成された蝸牛管のレーブグラフを示す。同図のごとく、渦巻状の構造が正しく表現されている。この例では穴の数ｇ＝０である。

【０２２８】この例の場合、まずはじめに図３１において太線で描かれている辺が生成された。つづいて辺 $E_1$ および $E_2$ が追加され、$E_3$ は拒絶された。なぜなら $E_3$ はグラフにおいて閉曲線を作るためである。同様の手順で $E_4$ が追加され、$E_5$、$E_6$、$E_7$、$E_8$ が拒絶された。処理はここで終了し、グラフは我々が予め知っている形状に一致した。

【０２２９】図３２は内耳の半規管に関するレーブグラフである。半規管は穴を３つ持っている。この場合、辺 $E_1$、$E_2$、…$E_8$ および $E_9$ が追加され、$E_{10}$、$E_{11}$、$E_{12}$ が拒絶された。その結果、グラフには正しく３つの閉曲線が生じた。

【０２３０】この自動化アルゴリズムが、通常人が目で行う方法よりも優れている理由は、十分な数の断面データがあるとき分岐が生じるような輪郭線に対してｑの値が大きくなるため、分岐を正しく扱うことができる点にある。この情報は、単に断面図を見ているだけでは得ることができない。現実に我々が実験を行ったとき、外部からの指示は全く不要であった。

【０２３１】（４）レーブグラフを用いた表面の符号化

一旦レーブグラフが構成されれば、表面の符号を得るこ

とは容易である。したがって、表面の符号化を直接行う代りに、レーブグラフまたは断面データを記述することによってオブジェクトを符号化することができる。符号はレーブグラフを上から下にスキャンすることにより、以下の手順で得ることができる。

【０２３２】１．ある高さにおいてグラフのノードが新たに現れたとき、対応する符号はPut_e2 である。このノードが既存の輪郭線（輪郭線＃ｎとする）に含まれていれば、対応する符号は Put_e2(n) である。

【０２３３】２．ある高さにおいてグラフのノードが消えたとき、対応する符号は Put_e0である。

【０２３４】３．２つの辺がマージされるとき、対応する符号は Put_el_merge である。新たな輪郭線の子輪郭線がマージされた２つの輪郭線の子輪郭線に一致しない場合、エラーが報告される。

【０２３５】４．辺が分れるとき、対応する符号は Put_el_divideである。

【０２３６】分離した輪郭線の子輪郭線（第２パラメータclist）は、輪郭線どうしの包含関係をチェックすることで決まる。分岐した一方の輪郭線（＃ｍとする）が他方（＃ｎとする）に含まれる場合、［１］で説明した第３パラメータは outsideとなる。それ以外（兄弟輪郭線）の場合、そのパラメータは inside となる。outside の場合、＃ｍの子輪郭線が＃ｎの子輪郭線になるかどうかがチェックされる。一方、inside の場合、＃ｍおよび＃ｎの子輪郭線が元の輪郭線の子輪郭線に一致するかどうかがチェックされる。

【０２３７】ある高さにおいて符号が計算された後、その高さにおける輪郭線どうしの親子関係が、対応する演算子を輪郭線の木構造に対して適応した結果得られるものと一致するかどうかがチェックされる。もし一致しなければ、エラーである。親子関係を破っている辺の１つの重さは０に修正され、自動化アルゴリズムが再実行される。

【０２３８】［実施形態］以上が前提技術である。この前提技術を本発明との関連を中心にまとめれば以下のとおりである。

【０２３９】［１］について

モース理論は特異点と指数の情報しか与えない。したがって、立体を構成するための胞体（セル）の種類は概念として判明するが、それを具体的にどのような幾何的関係をもって接続していくかは不明である。レーブグラフもモース理論の問題点を解決したわけではない。レーブグラフでは、特異点相互の関係はわかるが、結び目や階層関係は判明しない。結び目を知るためには、埋め込みに関する情報、すなわち幾何情報を知る必要がある。以上の問題点は、数学上の目的はともかく、立体を正しく符号化するという産業上の目的からすれば解決が必須である。

【０２４０】こうした観点から、輪郭線の木構造をもとに輪郭線どうしの階層関係を表現することができ、かつ立体の埋込み状態を視覚的に容易に指定することの可能なアイコン表示を導入した。このアイコン表示は立体を符号化するために必要かつ十分な４種類の演算子（Put_e0 など）と関連づけられているため、ユーザインタフェイスとしてはアイコンにより、またシステムの内部処理としては演算子により、立体の符号化を位相的な正しさを保証しながら行うことが可能となった。

【０２４１】この［１］の技術は、立体を新たに設計する場合を目的とする。したがって、既存の立体をその断面から自動的に再構成するものではない。一方、本発明はキーフレームを断面として立体を再構成することに等しい。したがって、［１］の技術は本発明の主たる手段にはならず、実施形態４においてひとつの応用例として利用される。

【０２４２】［２］について

［１］によって立体が符号化された状態を前提に立体の表面を生成する。［１］の符号化によれば、臨界断面はそれぞれ演算子（またはアイコン）に対応しているためすべて判明する。したがって、臨界断面の間をホモトピーによって接続すれば立体の表面を生成することができる。臨界断面の間で立体の位相が変化することはなく、一方、ホモトピーは立体の位相を変化させないためである。［２］の技術は［１］と関連しており、［１］同様に実施形態４で利用する。

【０２４３】［３］について

断面データが与えられたとき、対応輪郭線対の間で対応しあう点を検出し、それらの輪郭線の間の表面を生成する。［３］では基本的に輪郭線の分岐は考慮せず、対応輪郭線対は判明しているものとして、これらの輪郭線における対応点対を探索する。

【０２４４】対応点対の検出にはトロイダルグラフを用いる。本発明では従来より提案されている離散タイプのトロイダルグラフを用いてもよいが、本発明者が提案した連続タイプのトロイダルグラフを用いることができる。より自然な対応点対を検出するためには後者が有利である。

【０２４５】連続トロイダルグラフを用いるとき、本発明者の提案する「最多かつ最長のランを通過する経路」を見い出すアルゴリズムを用いてもよい。ランは計算量削減のために所定の最近点対をグループ化するもので、輪郭線対を結ぶ自然な経路を発見できるとともに、計算時間短縮の効果を併せもつ。

【０２４６】最近点対が判明すれば、それら対の間を補間することにより、一方の輪郭線ｆ（ｘ）上のすべての点と他方の輪郭線ｇ（ｙ）上のすべての点を対応づけることができる。ｘ、ｙをそれぞれの輪郭線の弧上の距離とすれば、対応点対はｙ＝Ｕ（ｘ）という関数Ｕで表現できる。Ｕが判明すれば、ｆ（ｘ）とｇ（Ｕ(x)）をホモトピーＦで結べば曲面が生成できる。

【０２４７】［３］の技術は［４］によって対応輪郭線対が判明した後、本発明の主たる手段として実施形態１、２で利用する。本発明ではアニメータの労力を軽減するためにキーフレームどうしの対応点の入力をアニメータに要求しないため、対応点対の自動検出に［３］を活用する。

【０２４８】［４］について

［３］の準備として、隣接する断面間における対応輪郭線対を検出する。対応輪郭線対の検出とレーブグラフの生成は同義と考えられる。

【０２４９】対応輪郭線対の検出には場合分けがある。まず、隣接する断面のそれぞれに含まれる輪郭線の数が等しい場合、重み関数に従って近い輪郭線どうしが対応づけられる。輪郭線の数が異なる場合には、オブジェクトの連結成分とハンドルの数を考慮しながら近い輪郭線どうしを関連づけていく。一連の処理は本発明者の提案するレーブグラフの自動生成アルゴリズムで実現できる。

【０２５０】ただし本発明では、ユーザによる指定を容易にするために、連結成分とハンドルの数に関する所定のデフォルト設定を行う。設定は、アニメーション製作を念頭においている。これらのパラメータのほか、

［２］で導入した輪郭線の階層構造を示す木構造をもとに自然消滅または自然発生すべき輪郭線の推定を行う。さらに、ユーザの意図を確認するための簡単な指示入力を受け付ける。

【０２５１】実施形態１．前提技術では、立体の表面をホモトピーによって構成したとき、そうして得られる立体がそのまま目的のオブジェクトとなった。したがって、前提技術ではオブジェクトの三次元形状を静止画として表現することを主眼とした。

【０２５２】一方、本発明では、立体は最終的にアニメーションのフレームを生成するための手段として用いられるにすぎず、オブジェクトの二次元形状を動画として表現することに主眼がある。このために、前提技術における立体のｚ軸を空間軸から時間軸に変更する。実施形態１ではまず、本発明に係るアニメーションの自動中割システムを説明する。

【０２５３】本実施形態を実現するシステムのハードウエアは、自動中割を行うためのプラットホームであるＰＣと、このＰＣに対してアニメータの描いたキーフレームの画像を入力するためのイメージスキャナだけでよい。キーフレームをＰＣ上の描画ソフトウエアなどで作成する場合にはイメージスキャナも不要である。これ以外の構成としては、実際に作成したアニメーションの各フレーム画像を出力するプリンタと、アニメーションに関するデータを記録する磁気記憶装置などがあればよい。

【０２５４】図３３は、本実施形態に係る中割システム２のソフトウエアモジュール構成図である。同図のごとくこのシステムは主に、キーフレームの画像を受け付けるキーフレーム受付部４、複数のキーフレームをもとにアニメーション立体を生成するアニメーション立体生成部６、生成されたアニメーション立体を切断して中間フレームの画像を取得する中割部２６からなる。

【０２５５】アニメーション立体生成部６は、隣接するキーフレーム間における対応輪郭線対を検出する対応輪郭線検出部８と、対応輪郭線対の間の表面をホモトピーによって生成する表面生成部２０からなる。

【０２５６】対応輪郭線検出部８は、前提技術［４］を利用する。対応輪郭線検出部８は、連結成分に関する関する情報を処理する連結成分関連処理部１０、輪郭線の木構造をもとに消滅する輪郭線の推定などを行う木構造関連処理部１２、連結成分および木構造を考慮したうえで輪郭線間の対応の可能性をそれらの近さをもとに評価する輪郭線対応評価部１４、評価の結果にもとづき対応する可能性が高い輪郭線対を表示するオプション機能である候補表示部１６をもつ。

【０２５７】一方、表面生成部２０は、対応輪郭線対の間の対応点対を前提技術［３］をもとに検出する対応点検出部２２と、［３］（４）「ホモトピーモデルに基づく表面の生成」を用いて拡張トロイダルグラフからアニメーション立体の表面を生成するホモトピー関連処理部２４をもつ。以下、中割システム２によって実際に中割を行う手順を図３４を用いて説明する。

（Ｓ２）キーフレームの入力

まず、ユーザは複数のキーフレームを入力する。キーフレームに描かれているオブジェクトの輪郭線は各種形状関数、例えば、点、円、多角形、ベジェ、ＮＵＲＢＳなどによって近似する。関数はユーザが選択可能としてもよい。精度が要求される場合は、ＮＵＲＢＳ曲線など二次曲線を忠実に再現できる関数が選ばれる。

【０２５８】ここでは図３５に示す２枚のキーフレームＦ０とＦ１が入力されたとする。これらのキーフレームには円形オブジェクト３０（以下「Ｏｃ３０」）と長方形オブジェクト３２（以下「Ｏｒ３２」）が描かれている。キーフレームＦ０で画面左にあったＯｃ３０がキーフレームＦ１では右に移動し、形状も円から楕円に変わっている。キーフレームはフリーハンドで描画すればよい。

【０２５９】（Ｓ４）パラメータの指定

つづいてユーザは各種パラメータを指定する。パラメータによる指定は以下の２つである。

【０２６０】１．オブジェクトの連結成分の数$N_{conn}$

２．輪郭線の木構造を考慮するか否か

連結成分とは、キーフレームに登場する物体の個数をいうが、ここでは２つのキーフレームにおける連結成分の大きいほうを$N_{conn}$と決めることにする。なお、ユーザが連結成分の数を指定しなかった場合、システムは以下のデフォルト設定を行う。

【０２６１】$N_{conn}$ ＝ｍａｘ（$N^{F0}$，$N^{F1}$）
ただし、$N^{F0}$、$N^{F1}$はそれぞれキーフレームＦ０、Ｆ１に含まれる輪郭線の数である。ここではユーザは指定をしなかったものとする。この場合、図３５から$N^{F0}$＝$N^{F1}$＝２であるから、$N_{conn}$＝２が設定される。この処理は図３３の連結成分関連処理部１２で行われる。
【０２６２】一方、木構造は前提技術［２］で親輪郭線、子輪郭線などのことばで定義した輪郭線どうしの包含関係である。木構造を考慮する場合、後の対応輪郭線対の検出の際、
・輪郭線が消えるとしたら木構造の末端にある子輪郭線（木構造のリーフ）である
・自然発生する輪郭線は既存の輪郭線の親輪郭線になれない
という規則が課される。なお、ユーザがこの指定をしなかった場合、システムはデフォルトとして、「木構造を考慮しない」を内部的に設定する。ここでもユーザは指定をしなかったものとし、木構造が考慮されないとする。この処理は図３３の木構造関連処理部１２で行われる。
【０２６３】（Ｓ６）輪郭線間の対応の評価
つぎに、輪郭線対応評価部１４において輪郭線間の対応の評価を行う。この評価の際、Ｓ４における指定の内容を考慮したうえで、キーフレームＦ０とＦ１の間で対応しあう可能性のある輪郭線対をすべて抽出する。いま、連結成分の数$N_{conn}$は２であるため、可能性のある輪郭線対を（Ｆ０に含まれる輪郭線，Ｆ１に含まれる輪郭線）と表記すれば、
組合せ１：（Ｏｃ３０，Ｏｒ３２）かつ（Ｏｒ３２，Ｏｃ３０）
組合せ２：（Ｏｃ３０，Ｏｃ３０）かつ（Ｏｒ３２，Ｏｒ３２）
の２組に限られる。仮に、
組合せ３：（Ｏｃ３０，Ｏｃ３０）かつ（Ｏｒ３２，Ｏｃ３０）
を考えると、２つの輪郭線が同一の輪郭線に集まることになり、連結成分が１個になって$N_{conn}$＝２に反する。したがって、組合せ１、２のみとなる。なお、木構造については考慮しないため、これら２つの組合せがそのまま対応輪郭線対の候補となる。
【０２６４】つづいて、組合せ１、２のいずれが妥当であるかを判定する。この判定は、組合せ１、２の含まれる合計４つの輪郭線対のそれぞれについてその近さを評価し、最も近いもの、すなわち最近輪郭線対を求める。最近輪郭線対は前提技術［４］の重み関数を用いて見い出すことができる。いまの場合、（Ｏｒ３２，Ｏｒ３２）が最近輪郭線対になるため、これを含む組合せ２が自動的に対応輪郭線対の組合せとして選択される。
【０２６５】（Ｓ８）対応輪郭線対の決定
Ｓ６で対応輪郭線対が一応見つけ出された。本システムでは、候補表示部１６によってＳ６で決まった対応関係を図３６のごとく画面に表示する。この対応関係が正しければユーザは「ＯＫ」を入力し、対応輪郭線対が決定される。ただし、Ｓ８はオプション機能として通常はスキップしてもよい。



【０２６６】（Ｓ１０）対応点対の検出
ここで処理は図３３の表面生成部２０に移る。まず、対応輪郭線対（Ｏｃ３０，Ｏｃ３０）および（Ｏｒ３２，Ｏｒ３２）のそれぞれについて、輪郭線上の対応点対を検出する。
【０２６７】（Ｏｃ３０，Ｏｃ３０）を例に説明する。図３７に示すごとく、２つの輪郭線を同じ大きさの正方形に収まるように正規化する。輪郭線の形状はスプライン曲線やＮＵＲＢＳ曲線で表され、上下の輪郭線の形状関数をそれぞれｆ（ｘ）とｇ（ｙ）とおく。ｘ、ｙはそれぞれ弧上の距離である。この表記は前提技術［３］（３）において、連続トロイダルグラフのパラメータに弧長をとる場合に対応する。
【０２６８】つづいて上下の輪郭線間の最近点対を求める。最近点対とは、対応輪郭線対Ａ、Ｂそれぞれの上の点ａ、ｂについて、点ａから見て輪郭線Ｂ上の最も近い点が点ｂであり、かつ点ｂから見て輪郭線Ａ上の最も近い点が点ａであるとき、これら点ａ、ｂの対をいう。図３８はこうして求められた最近点対ｘｉ－ｙｉを示している。実際のインプリメントでは、上述のごとく、最多かつ最長のランを発見するアルゴリズムを活用すればよい。
【０２６９】この後、補間計算によって最近点対以外の点どうしの対応を決める。例えば図３８の場合、上の輪郭線のｘ０とｘ１の弧上の中間点は下の輪郭線のｙ０とｙ１の弧上の中間点に対応するなどと考えればよい。輪郭線上のすべての点の対応関係が判明すれば、ｙ＝Ｕ（ｘ）なる一様増加関数が決まり、この関数によって対応点対が記述される。Ｏｒ３２についても同様の処理を行う。
【０２７０】（Ｓ１２）表面の生成
つづいて、ホモトピー関連処理部２４により、アニメーション立体の表面が生成される。このために、前提技術［３］（４）のホモトピーによる表面の生成技術を利用する。すなわち、ｙ＝Ｕ（ｘ）によって２つの輪郭線上のすべての点が対応づけられているため、ホモトピーがこれら対応点対をファイバーのようにつなぐ。例えば対応点対をすべて直線でつなぐ場合、直線ホモトピー、
$F(x,t)=(1-t)f(x)+tg(x)$
を用いればよい。ただし、正確にいえばこの式のｇ（ｘ）はｇ（Ｕ(x)）である。アニメーションの場合、立体の設計のような厳密さは求められないため、実際にはほとんどの場合が直線ホモトピーでカバーできると考えられる。
【０２７１】図３９はこうして生成されたアニメーショ

ン立体を示す図である。このアニメーション立体の上面がキーフレームＦ０、下面がＦ１に対応する。このアニメーション立体の特徴は、Ｏｃ３０とＯｒ３２に関する２つの立体からなることである。ここで連結成分$N_{conn}$＝２であるから、これらの立体は図３９では交わって見えても、本システムではこれらは互いに相手を素通りするような別々の立体として扱われる。

【０２７２】（Ｓ１４）中割の実行確認

アニメーション立体が生成された後、システムはユーザに対して中割を実行するかどうかを問い合わせる。ここでユーザが「ＹＥＳ」を入力するとシステムは次のＳ１６に進み、「ＮＯ」を入力するとＳ２０へ進む。

【０２７３】（Ｓ１６）中割

システムはまず、中割に必要なパラメータの入力をユーザに求める。パラメータの例は、キーフレームＦ０、Ｆ１の表示時刻、生成すべき中間フレームの数などである。キーフレームの表示時刻が指定されれば、２つのキーフレームの表示時刻の差を例えば１／３０秒で割ることより、ＮＴＳＣ規格に則って生成すべき中間フレームの数が判明する。逆に、生成すべき中間フレームの数が直接入力されれば、フレームの表示時刻に関する情報がなくとも中割を行うことができる。

【０２７４】図４０は中割の様子を示している。いまキーフレームＦ０、Ｆ１がそれぞれ表示時刻ｔ０、ｔ１に対応しているとすれば、表示時刻ｔ＝ａｔ０＋（１－ａ）ｔ１、ａ∈［０，１］の中間フレームは、アニメーション立体の縦軸を（１－ａ）：ａに内分する点を含むｘｙ平面上の断面形状で決まる。同図の場合、Ｏｃ３０とＯｒ３２が一部重なった状態の中間フレームが得られる。

【０２７５】（Ｓ１８）アニメーションの表示

アニメーションは上記のパラメータａを動かすことにより、キーフレームＦ０とＦ１の間の任意の中間フレームの画像を得て生成される。図４１は、生成された中間フレームが２枚の場合を示している。同図のごとく、アニメーション立体をｔ＝２／３ｔ０＋１／３ｔ１、およびｔ＝１／３ｔ０＋２／３ｔ１という平面で切断して２枚の中間フレームが得られる。これらを２枚のキーフレームの間において同図の上から順に表示すれば、Ｏｒ３２の前をＯｃ３０が横切るアニメーションが完成する。

【０２７６】Ｏｃ３０がＯｒ３２の前後いずれを通過するかは、Ｏｃ３０とＯｒ３２に関する画像データのいずれを先に中間フレームに描画するかによる。したがって、Ｏｃ３０がＯｒ３２の前を通るアニメーションを作りたければ、Ｏｒ３２の描画を先に行い、Ｏｃ３０をこれに上書きすればよい。このため、システムはユーザに対して「どちらのオブジェクトが遠くにあるか」を問い合わせてもよい。ユーザがＯｒ３２のほうが遠いと指定すれば、システムはＯｒ３２を先に処理することで所望のアニメーションが得られる。



【０２７７】なおシステムは、表示されたアニメーションがユーザの希望に沿うかどうかを問い合わせてもよい。ユーザが「ＮＯ」を入力したとき、Ｓ８の処理に戻って輪郭線どうしの対応関係をマニュアルで修正してもよい。対応する可能性のある輪郭線対が多数存在するとき、図３３の候補表示部１６が、それらの輪郭線対を対応の可能性の高い順に画面に色を変えて表示していってもよい。

【０２７８】（Ｓ２０）アニメーション立体の記憶

Ｓ１４で「ＮＯ」が入力された場合、システムは中割を行わずに図３９のアニメーション立体を記憶装置に記憶する。この際、アニメーション立体に必要な名前を付けておくことにより、アニメーションのデータベース化が可能となる。実際にアニメーションを再生するときには、記憶装置から必要なアニメーション立体を呼び出してＳ１６の中割を行い、これを表示すればよい。

【０２７９】アニメーション立体の記憶に必要なデータの種類は以下のとおりである。

【０２８０】１．キーフレームの画像

２．キーフレーム間の対応輪郭線対の表示

３．対応輪郭線対における対応点対の表示

４．対応点対を接続するホモトピー

この中で４については、直線ホモトピーなどをデフォルトとして決めておけば不要になる。したがってデータ量も少なく、アニメーション立体は通信にも好適である。例えばアニメーション提供者側の送信装置によってアニメーション立体を各家庭に送り、各家庭側では受信装置でこれを受信する。受信装置はこのアニメーション立体に対してＳ１６の中割、およびＳ１８の表示を行えばよい。その場合、アニメーション立体の送り手の側で、生成すべき中間フレームの数など、中割に必要な情報をアニメーション立体に付加して送ってもよい。

【０２８１】実施形態２．実施形態１ではキーフレーム間で輪郭線の数が変化しない例を説明した。実施形態２ではこれが変化する例を用いて、パラメータの指定と輪郭線の発生、消滅の推定との関係を説明する。

【０２８２】例１．重なり合わない２つの円（タイプ１）

図４２は２枚のキーフレームＦ０、Ｆ１の画像を示している。この例では、キーフレームＦ０に１個のオブジェクトＯｃ４０、Ｆ１に２個の重なり合わないオブジェクトＯｃ４２、４４がある。以下、実施形態１の主要なステップに即して中割処理を説明する。

【０２８３】（Ｓ４）パラメータの指定

１．オブジェクトの連結成分の数$N_{conn}$

デフォルト値が$N_{conn}$＝ｍａｘ（$N^{F0}$，$N^{F1}$）で計算される。ここで$N^{F0}$＝１、$N^{F1}$＝２であるから、$N_{conn}$＝２が設定される。

２．輪郭線の木構造を考慮するか否か

デフォルトとして、「木構造を考慮しない」が内部的に

設定される。

【０２８４】（Ｓ６）輪郭線間の対応の評価

連結成分の数$N_{conn}$は２であるため、キーフレームＦ０とＦ１の間で対応しあう可能性のある輪郭線対は、

組合せ１：（Ｏｃ４０，Ｏｃ４２）かつＯｃ４４は自然発生

組合せ２：（Ｏｃ４０，Ｏｃ４４）かつＯｃ４２は自然発生

の２組に限られる。このうち、（Ｏｃ４０，Ｏｃ４２）が最近輪郭線対であるから、これを含む組合せ１が自動的に対応輪郭線対の組合せとして選択される。

【０２８５】この後、Ｓ８、Ｓ１０は実施形態１同様に進む。

【０２８６】（Ｓ１２）表面の生成

図４３は生成されたアニメーション立体を示す図である。Ｏｃ４０とＯｃ４２が結ばれ、Ｏｃ４４は任意の箇所で発生する。Ｏｃ４４に関する立体の頂点４６を含む断面が臨界断面になる。システムはこの臨界断面を例えばｔ＝（ｔ０＋ｔ１）／２の位置におく。なお、ここではＯｃ４４に関する立体を円錐として描いたが、これは円柱などでもよい。

【０２８７】処理はこの後、Ｓ１４、Ｓ１６と進んだとする。

【０２８８】（Ｓ１６）中割

実施形態１同様、アニメーション立体をｔ＝２／３ｔ０＋１／３ｔ１、およびｔ＝１／３ｔ０＋２／３ｔ１という平面で切断して２枚の中間フレームを得る。

【０２８９】（Ｓ１８）アニメーションの表示

図４４は、生成された中間フレームが２枚の場合を示している。同図のごとく、Ｏｃ４０は徐々にＯｃ４２になる一方、Ｏｃ４４が画面に次第に現れる。図４３でＯｃ４４に関する立体を円柱とすれば、Ｏｃ４４は突然最終的な大きさで画面に現れることになる。また、Ｏｃ４４に関する立体を円錐とし、その円錐の頂点を右側にずらせば、Ｏｃ４４が画面右側から次第に近づいてくるようなアニメーションになる。なお、Ｓ２０については実施形態１同様である。

【０２９０】例２．重なり合わない２つの円（タイプ２）

この例でも図４２の２枚のキーフレームＦ０、Ｆ１が入力されたとする。ただし今回は、Ｏｃ４４が自然発生するのではなく、Ｏｃ４０がＯｃ４２とＯｃ４４に分裂するアニメーションを作りたいとする。このために、以下の２つのアプローチが考えられる。

【０２９１】［アプローチ１］例１同様、Ｓ４でパラメータの指定をすることなくアニメーションを自動作成する。その場合、当然図４４に示すアニメーションができる。このアニメーションはユーザの希望に反するため、Ｓ１８においてユーザは「ＮＯ」を入力する。システムはＳ８の処理に戻り、ユーザは輪郭線どうしの対応関係をマニュアルで修正する。ユーザは明示的にＯｃ４０とＯｃ４２、およびＯｃ４０とＯｃ４４を連続的にマウスでクリックするなどの方法が考えられる。

【０２９２】［アプローチ２］Ｓ４で連結成分の数$N_{conn}$を明示的に指定する。この場合、$N_{conn}$＝１とすればよい。このときアニメーション立体は図４５のように二股に分岐する。分岐する際に鞍点４８が生じ、この鞍点４８を含む断面が臨界断面になる。システムはこの臨界断面を例えばｔ＝（ｔ０＋ｔ１）／２の位置におき、これをキーフレームのように扱う。すなわち、臨界断面上のアニメーション立体の２つの輪郭線５０、５２と、もとのキーフレームＦ０、Ｆ１のそれぞれの間でホモトピーを用いた表面の生成を行う。図４６は最終的に得られるアニメーションを示している。

【０２９３】例３．重なり合う２つの円（タイプ１）

図４７は例３のキーフレームＦ０、Ｆ１を示している。キーフレームＦ０では内側にＯｃ６０、外側にＯｃ６２という重なり合う円があり、キーフレームＦ１にはひとつの円Ｏｃ６４がある。

【０２９４】ここでもＳ４でパラメータを指定しないものとし、$N_{conn}$＝２、および「木構造を考慮しない」が設定される。キーフレームＦ０とＦ１の間で対応しあう可能性のある輪郭線対は、

組合せ１：（Ｏｃ６０，Ｏｃ６４）かつＯｃ６２は自然消滅

組合せ２：（Ｏｃ６２，Ｏｃ６４）かつＯｃ６０は自然消滅

の２組に限られる。このうち、（Ｏｃ６０，Ｏｃ６４）が最近輪郭線対であるから、これを含む組合せ１が選択される。

【０２９５】図４８は生成されたアニメーション立体を示す図である。このアニメーション立体は、外円Ｏｃ６２に関する立体と、内円Ｏｃ６０に関する立体の２つから構成される。これらの立体は図面上では重なっているが、$N_{conn}$＝２であるからシステムでは別個の立体として扱われる。図示しないが、こうして生成されたアニメーション立体に対して中割を行えば、当然ながら途中で外円Ｏｃ６２が突然消滅するアニメーションが得られる。

【０２９６】例４．重なり合う２つの円（タイプ２）

図４９は例４のキーフレームＦ０、Ｆ１を示している。キーフレームＦ０は例３と全く同じである。キーフレームＦ１にはひとつの長方形Ｏｒ７０がある。ここではキーフレームＦ０がトーラスを正面から見たところ、キーフレームＦ１が同じトーラスを９０°横から見たところであり、ユーザはトーラスが９０°回転するアニメーションを作りたいものとする。

【０２９７】ここで、Ｓ４でパラメータを指定しないとすれば、例３同様キーフレームＦ０とＦ１の間で対応しあう可能性のある輪郭線対は、

組合せ１：（Ｏｃ６０，Ｏｒ７０）かつＯｃ６２は自然消滅
組合せ２：（Ｏｃ６２，Ｏｒ７０）かつＯｃ６０は自然消滅
の２組に限られる。ここで仮に、計算上（Ｏｃ６０，Ｏｒ７０）が最近輪郭線対となれば、例３同様、Ｏｒ６２が途中で突然消滅するようなアニメーションが生成される。これはトーラスの回転とはまったく異なる映像である。この問題を解決するには以下の２つのアプローチが考えられる。

【０２９８】［アプローチ１］例２のアプローチ１同様の措置をとる。

【０２９９】［アプローチ２］Ｓ４で連結成分の数$N_{conn}$を明示的に１とする。このときアニメーション立体は図５０のようになる。このアニメーション立体では、トーラスの内円Ｏｃ６０が尾根線（リッジ）７２で消滅している。このリッジを含む断面が臨界断面であるる。最終的に生成されるアニメーションは図５１に示すとおりである。この例を変形すれば、人が目を閉じるようなアニメーションを作ることができるし、そのフレームを逆に辿れば目が開くアニメーションも作れる。

【０３００】さらに、この例ではトーラスにおける円Ｏｃ６０とＯｃ６２が階層構造をなすことから、別のアプローチも考えられる。

【０３０１】［アプローチ３］Ｓ４で明示的に「木構造を考慮する」と指定する。このとき、「輪郭線が消えるとしたら木構造の末端にある子輪郭線である」という原則が適用される。いまＯｃ６０、Ｏｃ６２それぞれの輪郭線を＃６０、＃６２と表記するとき、キーフレームＦ０の輪郭線の木構造は、
＃０－＃６２－＃６０
であるから、消滅する輪郭線があればそれは＃６０と推定できる。このため、アニメーション立体はやはり図５０のようになり、アプローチ２同様のアニメーションが得られる。したがって、連結成分の数に関する指定と木構造に関する指定は、一方が指定されたとき、他方に関係なく指定された処理をなすようシステムを設計すればよい。ユーザは、連結成分または木構造のうち、ケースバイケースで理解が容易なほうに関する指定を行うことができる。

【０３０２】例３と例４の相違は、アニメーションで表現しようとする対象が三次元物体として実存しうるか否かにある。例３のアニメーション立体（図４８）は実在しえない。一方、例４のアニメーション立体（図５０）は、実在のトーラスを回転させながら生成される立体と把握できる以上、実在しうる。実在する三次元物体を対象とするアニメーションを作成する場合、システムのデフォルト値として「木構造を考慮する」としてもよい。

【０３０３】以上、実施形態２ではキーフレーム間で立体の数が変化するような場合を考えた。ここではオブジェクトの形状として円を中心に説明したが、円と多角形は位相同形であるため、同じ原理でいろいろなオブジェクト形状に対応することができる。また、複雑なアニメーションであっても、各部は上述の４つのパターンのいずれかで構成されていることが多いため、本実施形態の応用範囲は広い。

【０３０４】実施形態３．実施形態１、２に沿ってシステムを設計する場合、オプション機能として設けたほうがよい技術を説明する。

【０３０５】（１）曲線セグメントの処理機能
図５２はこの機能を説明するためのキーフレームＦ０、Ｆ１を示す図である。キーフレームＦ０には２つの端点８２、８４をもつひとつの開曲線（曲線セグメント）８０が描かれ、キーフレームＦ１にはひとつの閉曲線８６が描かれている。

【０３０６】一方、実施形態２の例４は人の目が開く場合に応用できることを述べた。図５２のキーフレームは人の目が開く状態と考えることができる。したがって、実施形態２の例４の処理により、図５３に示すようなアニメーションを作成することができる。

【０３０７】しかしながら、キーフレームＦ０からキーフレームＦ１への変化としては、図５４に示すように、一方の端点８２から曲線セグメントが延びていき、他方の端点８４に到達する場合も考えられる。こうした場合に対応すべく、システムは両端が閉じるような開曲線の指定を受け付けるものとする。具体的には、システムはキーフレームに開曲線が存在するとき、図５３および図５４を表示してユーザの選択を促す。ユーザが入力しない場合は、デフォルトとして図５３が選択されたものとして通常の処理を行う。

【０３０８】図５５はこの曲線セグメント処理機能の効果を説明するもので、キーフレームＦ０は火山を横から見たところ、キーフレームＦ１はその火山を上から見たところを示している。キーフレームＦ０は稜線９０と、冠雪線９２という２つの開曲線からなる。ここで、稜線９０についてはデフォルト設定、冠雪線９２については端点が延びてつながるという指定をすれば、キーフレームＦ０からキーフレームＦ１までの自然な視点変更アニメーションを作成することができる。

【０３０９】（２）視点移動に対する補正機能
例えばカメラが平行に移動したり、景色を見る人が列車に乗っている場合など、画面上のオブジェクトがすべて同じ方向に平行移動する場合がある。

【０３１０】図５６では２つのキーフレームＦ０、Ｆ１に長方形Ｏｒ１００と円Ｏｃ１０２が描かれているが、これらはキーフレーム間で平行移動している。そのため、このまま重み関数によって対応輪郭線対を検出した場合、（Ｏｃ１０２，Ｏｒ１００）が最近輪郭線対として誤って検出されうる。その結果、ユーザが予期しないアニメーションが生成される。

【０３１１】そこで本システムは、複数のオブジェクトがキーフレーム間で同じような移動をする場合、ユーザからその旨の指定を受け付ける。ユーザが「平行移動を考慮する」を指定した場合、システムは既存の画像領域マッチング手法などによってオブジェクトの動きベクトルを検出した後、オブジェクトの動きをゼロに戻して対応輪郭線対の検出を行う。

【０３１２】実施形態４．実施形態１～３では、前提技術［３］［４］をベースとしてシステムを説明した。これらのシステムは、ユーザに位相幾何学的な知識を要求しない点に特徴をもつ。

【０３１３】一方、位相幾何学に関する知識はなくとも、ある程度簡単なアニメーションについてはアニメーション立体を頭の中で想像できる場合もある。そうした場合、前提技術［１］［２］をベースとする新たなシステムの可能性が開ける。すなわち、ユーザがアニメーション立体を自らアイコンによって作り上げていくようなシステムである。アイコンによって立体を設計する場合、前提技術［１］［２］により、立体の位相の正しさが保証される。したがって、いったんアニメーション立体が作成できれば、中割によって生成される中間フレームも必ず位相的に正しいものとなる。実施形態４のシステムは、実施形態１～３のシステムに対する補助システムとしてインプリメントすればよい。

【０３１４】図５７は、実施形態４の中割システム１２０を実現するソフトウエア構成図である。同図において図３３と同等の部材には同一の符号を与えて説明を省略する。

【０３１５】本システムのアニメーション立体生成部１３０は、アニメーション立体の骨格に相当する部分を生成する骨格生成部１３２と、骨格にパッチを貼り付けて表面を生成する表面生成部１５０をもつ。

【０３１６】骨格生成部１３２は、アイコン処理部１３４を含む。アイコン処理部１３４は、アイコンユーザインタフェイス部１３６（以下、ユーザインタフェイスをＵＩと表記）と接続検査部１３８を含む。アイコンＵＩ部１３６は、レーブグラフのアイコン表示をＵＩとして提供する、レーブグラフに関するエディターである。アイコンＵＩ部１３６は、ｋセルに対応したアイコンやダミーアイコンなどを準備し、これらを画面に表示することによってユーザの選択を待つ。ユーザはクリック・アンド・ドラッグによって必要なアイコンを必要な位置に置いていくことができる。

【０３１７】一方、接続検査部１３８は、アイコンどうしの間に不正な接続関係がないかどうかを判定する。例えば、アニメーション立体の頂点にいきなり演算子 Put_e0に対応するアイコンがくることはない。また、各演算子に対してつぎに接続可能な演算子が決まるため、その対応関係に該当しないアイコンの列びがあればエラーを表示してユーザに通知する。

【０３１８】表面生成部１５０は、ガイディング曲線処理部１５２とホモトピー関連処理部１５４を含む。ガイディング曲線処理部１５２は、ガイディング曲線の入力を受け付ける。前提技術［２］ではホモトピーＦとして（ｉ）～（ｖ）を導入し、そのうち（ｖ）をユーザが自由に指定するガイディング曲線と位置づけたが、実際のインプリメンテーションでは、ホモトピーを決めるという意味において、（ｉ）～（ｉｖ）のすべてをガイディング曲線と考えることができる。ガイディング曲線を（ｉ）の直線ホモトピーとすると、対応輪郭線対の対応点対を画面上で指定するとその対応点対が直線で結ばれる。ガイディング曲線が（ｉｖ）のカージナルスプラインの場合、スプライン曲線を決めるために４点の指定が必要である。したがって、隣接する４つの輪郭線について対応点を指定することにより、それら４つの輪郭線の区間に関するガイディング曲線が決まる。ガイディング曲線処理部１５２では、ユーザによる対応点対の指定に従い、ガイディング曲線の式を計算して決めていく。

【０３１９】ホモトピー関連処理部１５４は、実施形態１のごとく、輪郭線上のすべての点が隣接する輪郭線上に対応する点をもつ場合、実施形態１同様にそれらの輪郭線間をファイバーのように接続する。しかし、ユーザの指定した対応点対のみから表面を生成する場合は、対応輪郭線対と隣接するガイディング曲線によって形成される領域ごとに表面パッチを当てるものとする。図５８は表面パッチを貼り付ける方法を説明する図である。同図では、対応輪郭線対の形状関数をｆとｇ、ガイディング曲線の形状関数をＧ０とＧ１としている。このとき、同図の点Ｐのごとく、Ｇ０とＧ１をα：（１－α）に内分し、ｆとｇをβ：（１－β）に内分する点の位置は、

（１－β）ｆ＋βｇ

という曲線と、

（１－α）Ｇ０＋αＧ１

という曲線のブレンディング（混ぜ合わせ）として決めることができる。ブレンディングの例として、

【数２０】（１－β）ｆ（α）＋βｇ（α）＋（１－α）Ｇ０（β）＋αＧ１（β）－（１－β）（１－α）ｆ（０）－β（１－α）ｆ（１）－（１－β）αｇ（０）－βαｇ（１）

などを用いることができる。こうしてこの領域の表面パッチが一意的に定まる。他の領域についても同様である。

【０３２０】以下、この構成による中割の手順を図５９のフローチャートを用いて説明する。

【０３２１】まずユーザは、生成したいアニメーションから逆にアニメーション立体の形を思い浮かべ、その立体をアイコンによって表現する（Ｓ３０）。いま、円が２つに分裂するアニメーションを作りたいとすれば、作成すべきアニメーション立体は図６０のような二股に分岐した立体である。そこでユーザはこれをアイコンによ

って図６１のように入力する。同図では、演算子 Put_e2、Put_e1_divide、および２つの Put_e0 に相当するアイコンがこの順に置かれている。

【０３２２】つづいて、システムはアイコンの入力が完了した時点で、キーフレームを順に入力するようユーザに指示する。本実施形態では、フレーム上でオブジェクトがとる形状が時間の経過とともに位相同形でなくなるような臨界的なフレームをキーフレームとして入力する。図６１の場合、$e^2$セルの生成に相当するフレーム、$e^1$の貼り付けに相当するフレーム、$e^0$の貼り付けに相当するフレームがそれぞれキーフレームである。

【０３２３】ユーザはシステムからの指示に従い、ｔ＝ｔ０、ｔ＝（ｔ０＋ｔ１）／２、ｔ＝ｔ１の３つのキーフレームの画像を入力する（Ｓ３２）。図６２はこのときの画面表示を示している。なお、ｔ＝ｔ０とｔ＝ｔ１のキーフレームは平らな頂上と底部に当たるため、前提技術［２］の「縮退のある場合」の考慮を行う。

【０３２４】ここでシステムはガイディング曲線の入力待ちとなる。ユーザはマウスなどによってガイディング曲線を入力し、対応輪郭線対の対応点対の位置をシステムに伝える（Ｓ３４）。なお、図６２の場合、ｔ＝ｔ０とｔ＝（ｔ０＋ｔ１）／２の間に Put_e1_divide のための道ｃ（ｔ）を定義しなければならないため、これも他のガイディング曲線とともに与える。ガイディング曲線の入力が終了したときの様子は図６３に示されている。

【０３２５】つぎに、ホモトピー関連処理部１５４が必要な表面パッチを貼り付けることでアニメーション立体の表面を生成する（Ｓ３６）。以降の処理は図３４のＳ１４以降と同じである。

【０３２６】以上が本実施形態による自動中割技術を用いたアニメーションの作成方法である。

【図面の簡単な説明】

【図１】　輪郭線の不当な変換例を示す図である。

【図２】　特異点の指数、ｋ次元セルおよびそれによって符号化される物体の関係を示す図である。

【図３】　（ａ）～（ｃ）はそれぞれが同じモースの指数の配列をもつ３組の曲面を示す図である。

【図４】　（ａ）～（ｃ）はトーラスとそのレーブグラフの関係を示す図である。

【図５】　（ａ）（ｂ）は輪郭線の親子関係、およびその木構造による表現を示す図である。

【図６】　演算子を用いてトーラスを符号化する方法を示す図である。

【図７】　疑似パスカルコードによって演算子のプログラミング例を示す図である。

【図８】　疑似パスカルコードによって演算子のプログラミング例を示す図である。

【図９】　疑似パスカルコードによって演算子のプログラミング例を示す図である。

【図１０】　それぞれがセルに対応するアイコンを示す図である。

【図１１】　セルの貼り合わせを示す図である。

【図１２】　オブジェクトのレーブグラフをアイコンによって示した図である。

【図１３】　オブジェクトの断面の輪郭線を示す図である。

【図１４】　オブジェクトを構成するための演算子を示す図である。

【図１５】　輪郭線のホモトピー変形を示す図である。

【図１６】　演算子を構成する主な４つの要素を示す図である。

【図１７】　ガイディング曲線によって上の輪郭線が徐々に下の輪郭線に変形される様子を示す図である。

【図１８】　微分不可能な点を含む物体を示す図である。

【図１９】　図１８の物体を微分可能な形状関数をもつ物体に置き換えた様子を示す図である。

【図２０】　頂部も分岐部もそれぞれ水平面である物体を示す図である。

【図２１】　図２０の物体の高さを０として表現した状態を示す図である。

【図２２】　物体のリッジを示す図である。

【図２３】　火山のリム構造を示す図である。

【図２４】　従来の三角形手法によって人工的な皺が生じた様子を示す図である。

【図２５】　最長かつ最多のランを得るためのアルゴリズムによって最終的に見いだされたランを示す図である。

【図２６】　連続トロイダルグラフにおいて、ｎｎ＝２のときの経路を示す図である。

【図２７】　最近点対以外の点の対応を線形補間によって導いた様子を示す図である。

【図２８】　図２７の経路に従って生成された表面を示す図である。

【図２９】　オブジェクトの分岐箇所を示す図である。

【図３０】　（ａ）～（ｅ）は輪郭線どうしの対応の態様を示す図である。

【図３１】　断面データを基に構成された蝸牛管のレーブグラフを示す図である。

【図３２】　内耳の半規管に関するレーブグラフを示す図である。

【図３３】　実施形態１に係る中割システムのソフトウエアモジュール構成図である。

【図３４】　実施形態１の中割システムによって実際に中割を行う手順を示すフローチャートである。

【図３５】　実施形態１のキーフレームＦ０とＦ１を示す図である。

【図３６】　実施形態１のＳ６で決まった対応関係を示す図である。

【図３７】　実施形態１の対応点の検出（Ｓ１０）のために輪郭線を正規化する様子を示す図である。

【図３８】　実施形態１の対応点の検出（Ｓ１０）によって求められた最近点対を示す図である。

【図３９】　実施形態１で生成されたアニメーション立体を示す図である。

【図４０】　実施形態１の中割の様子を示す図である。

【図４１】　実施形態１で生成された中間フレームの数が２の場合のアニメーションを示す図である。

【図４２】　実施形態２の例１のキーフレームＦ０、Ｆ１の画像を示す図である。

【図４３】　実施形態２の例１について生成されたアニメーション立体を示す図である。

【図４４】　実施形態２の例１について生成されたアニメーションのフレーム画像を示す図である。

【図４５】　実施形態２の例２について生成されたアニメーション立体を示す図である。

【図４６】　実施形態２の例２について生成されたアニメーションのフレーム画像を示す図である。

【図４７】　実施形態２の例３のキーフレームＦ０、Ｆ１の画像を示す図である。

【図４８】　実施形態２の例３について生成されたアニメーション立体を示す図である。

【図４９】　実施形態２の例４のキーフレームＦ０、Ｆ１の画像を示す図である。

【図５０】　実施形態２の例４について生成されたアニメーション立体を示す図である。

【図５１】　実施形態２の例４について生成されたアニメーションのフレーム画像を示す図である。

【図５２】　実施形態３の機能（１）を説明するためのキーフレームＦ０、Ｆ１の画像を示す図である。

【図５３】　図５２のキーフレームから実施形態２の例４の技術によってアニメーションを作成したときの様子を示す図である。

【図５４】　図５２のキーフレームから実施形態３の機能（１）を用いてアニメーションを作成したときの様子を示す図である。

【図５５】　実施形態３の機能（１）の効果を説明する図である。

【図５６】　実施形態３の機能（２）を説明するためのキーフレームＦ０、Ｆ１の画像を示す図である。

【図５７】　実施形態４の中割システム１２０を実現するソフトウエア構成図である。

【図５８】　ホモトピー関連処理部によって表面パッチを貼り付ける方法を説明する図である。

【図５９】　実施形態４の中割の手順を示すフローチャートである。

【図６０】　実施形態４でユーザが生成すべきアニメーション立体を示す図である。

【図６１】　図６０のアニメーション立体のアイコン表現である。

【図６２】　図６０のアニメーション立体のキーフレームの画像と位置を示す図である。

【図６３】　図６２のキーフレーム間にガイディング曲線をつけた状態を示す図である。

【符号の説明】

２，１２０　中割システム、４　キーフレーム受付部、６，１３０　アニメーション立体生成部、８　対応輪郭線検出部、１０　連結成分関連処理部、１２木構造関連処理部、１４　輪郭線対応評価部、１６　候補表示部、２２　対応点検出部、２４　ホモトピー関連処理部、２６　中割部、１３２　骨格生成部、１３４　アイコン処理部、１３６　アイコンＵＩ部、１３８　接続検査部、１５０表面生成部、１５２　ガイディング曲線処理部、１５４　ホモトピー関連処理部。

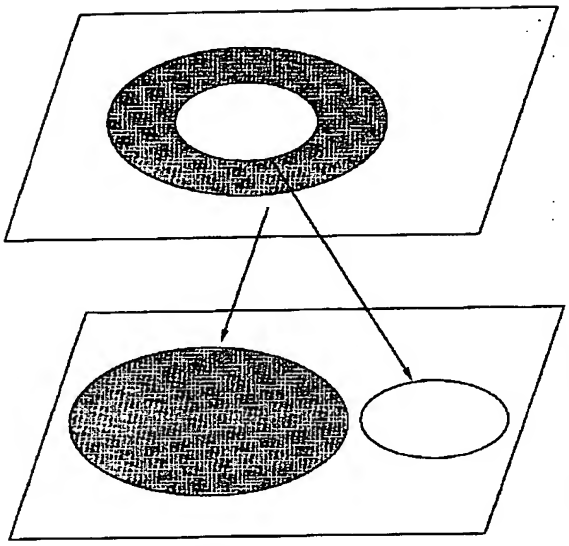
【図７】

```
program operators(input, output);
constant
        enabled = true;
        disabled = false;
        inside = true;
        outside = false;
        end_of_list = -1;
type
        contour_number = 0..max_contour_number;
        child_list = array[1..maxchildren] of contour_number;
        pointer_to_child_list = ↑ child_list;
var
        children: array[contour_number] of pointer_to_child_list;
        parent#: array[contour_number] of contour_number;
        number of children: array[contour_number] of integer;
        most_recently_created#: contour_number;
        contour_status: array[contour_number] of boolean;
```

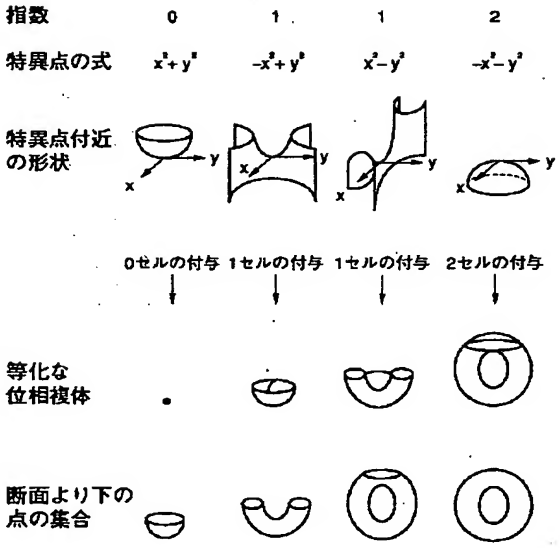
【図８】

```
procedure add_listed_children(n:contour_number;clist:pointer_to_child_list);
        {details are omitted}
procedure remove_listed_children(n:contour_number;clist:pointer_to_child_list);
        {details are omitted}
function are_children(n:contour_number;clist:pointer_to_child_list);boolean;
        {details are omitted}
function in_list(n:contour_number;clist:pointer_to_child_list);boolean;
        {details are omitted}
function list_containing_only(n:contour_number):pointer_to_child_list;
var     n_as_list: pointer_to_child_list;
begin
        new(n_as_list);
        n_as_list ↑[1]:= n;
        n_as_list ↑[2]:= end_of_list;
        list_containing_only:= n_as_list;
end;
```

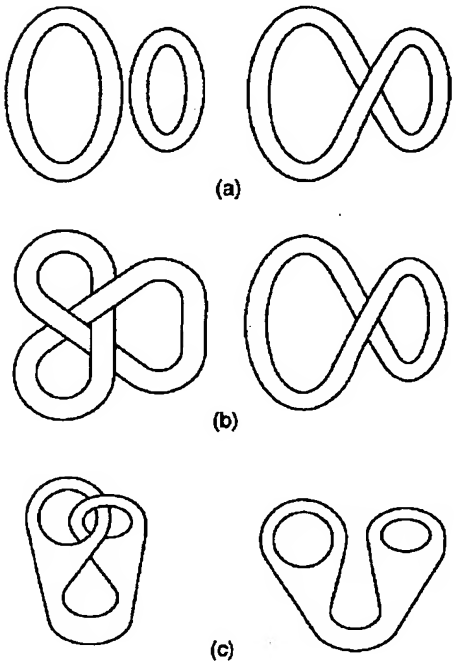
【図１】



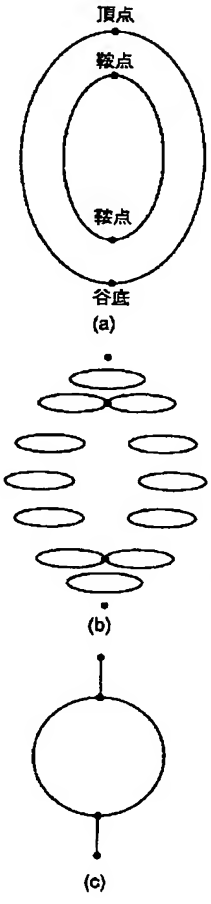
【図２】



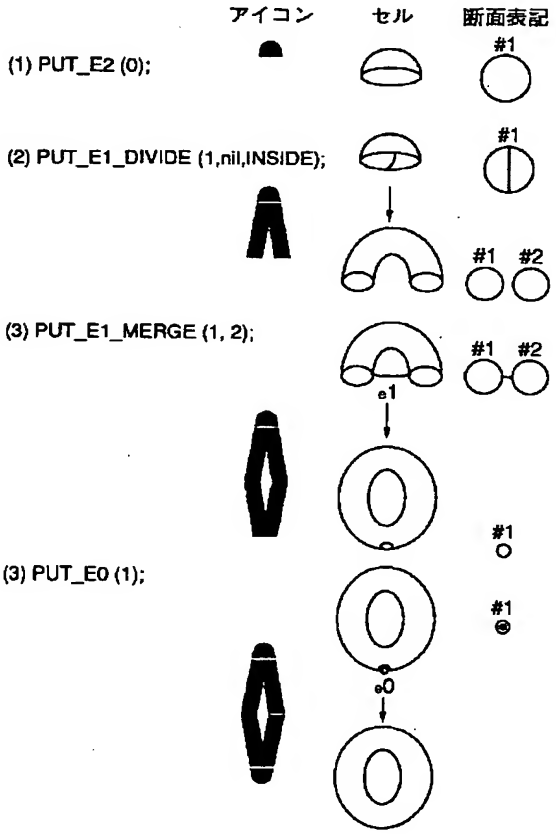

【図３】



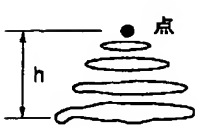

【図４】



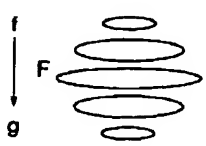

【図６】



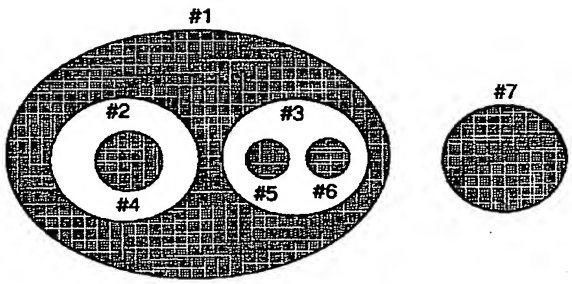

【図１６】



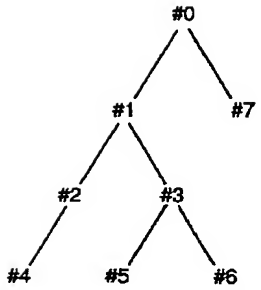

【図１９】



【図５】




(a)

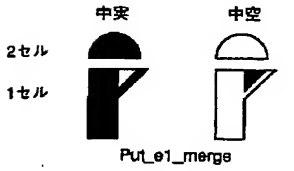



(b)

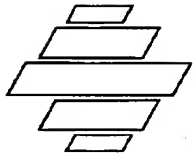
【図９】

```
a
procedure put_e2(n: contour_number);
begin
        if (contour_status[n] = disabled) then go to error;
        create_new_contour;
        add_listed_children(n,list_containing_only(most_recently_created#));
end;
b
procedure put_e0(n: contour_number);
begin
        if ((contour_status[n] = disabled) or not all_successor_disabled(n))
                then goto error;
        contour_status[n]:= disabled;
end;
c
procedure put_e1_divide(n:contour_number); clist: pointer_to_child_list; insideboolean);
begin
        if ((contour_status[n] = disabled) or (contour_status[parent#[n]]=disabled))
                then goto error;
        create_new_contour;
        add_listed_children(most_recently_created#, clist);
        if(not inside and are_children(parent#[n], clist)
        and not in_list(n, list)) or (clist = nil)) )
                then begin
                    remove_listed_children(parent#[n], clist);
                    add_listed_children(n,list_containing_only(most_recently_created#));
                end
        else if (inside and(are_children(n, clist) or (clist = nil)))
                then begin
                    remove_listed_children(n, clist);
                    add_listed_children(parent#[n],list_containing_only(most_recently_created#));
                end
        else go to error;
end;
d
procedure put_e1_merge(c1:contour_number; c2:contour_number);
begin
        if ((contour_status[c1] = disabled) or (contour_status[c2] = disabled))
                then goto error;
        if (c1 = parent#[c2]) then
                add_listed_children(parent#[c1], children[c2]);
        else if (parent#[c1] = parent#[c2] then
                add_listed_children(c1, children[c2]);
        else go to error;
        remove_listed_child(parent#[c2], list_containing_only(c2));
        contour_status[c2]:= disabled;
end;
```

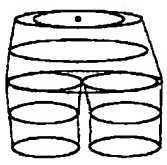
【図１０】



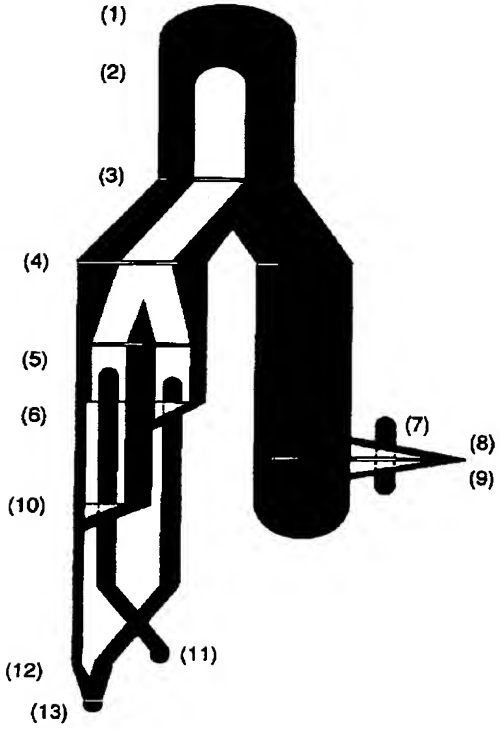

【図１８】



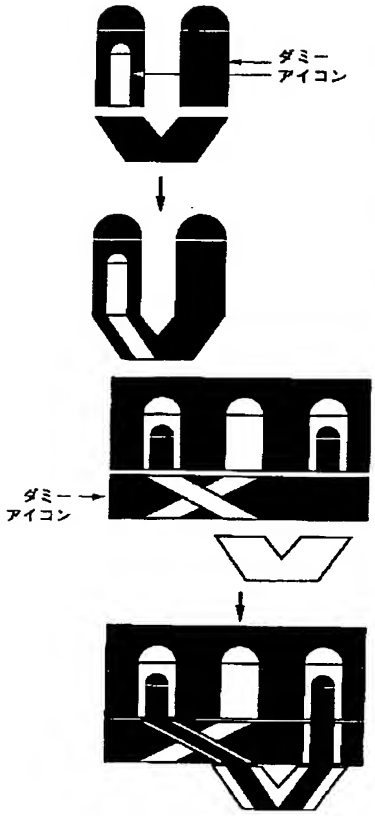
【図２０】



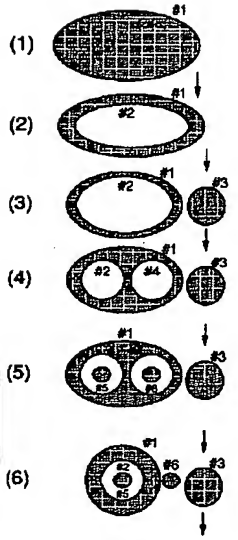
【図１２】



【図１１】



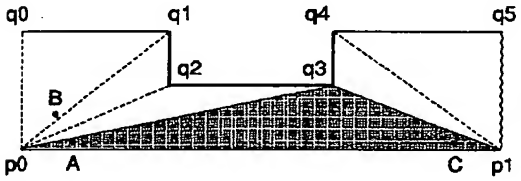

【図１３】



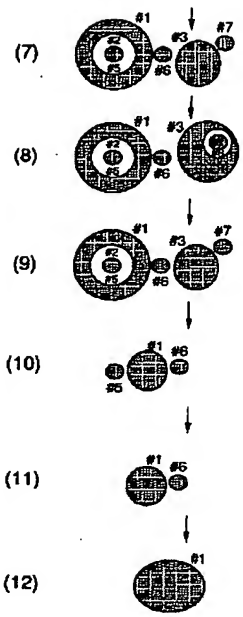

【図１４】

```
1. PUT_E2(0);
2. PUT_E2(1);
3. PUT_E1_DIVIDE(1, nil, INSIDE);
4. PUT_E1_DIVIDE(2, nil, INSIDE);
5. PUT_E2(2); PUT_E2(4);
6. PUT_E1_MERGE(1, 4);
7. PUT_E2(0);
8. PUT_E1_DIVIDE(3, list_containing_only(7), OUTSIDE);
9. PUT_E1_MERGE(3, 8); PUT_E0(7); PUT_E0(3);
10. PUT_E1_MERGE(1, 2);
11. PUT_E0(5);
12. PUT_E1_MERGE(1, 6);
13. PUT_E0(1);
```

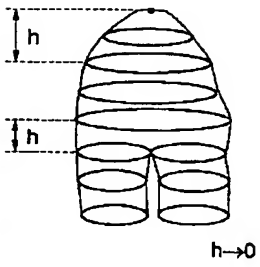
【図２４】



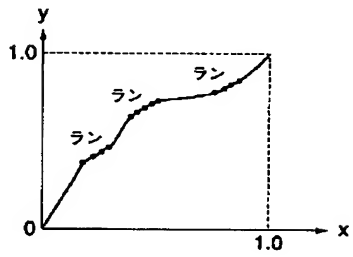

【図１５】

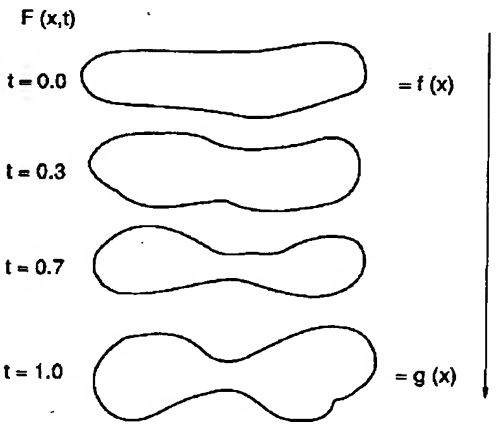



【図２１】

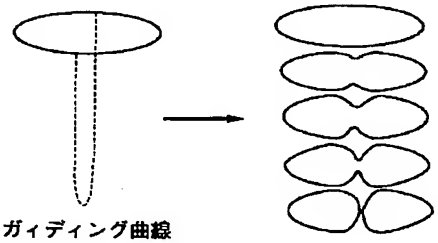



【図２５】

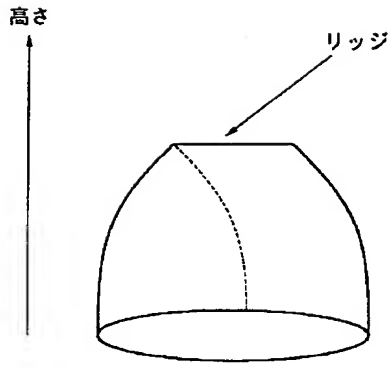


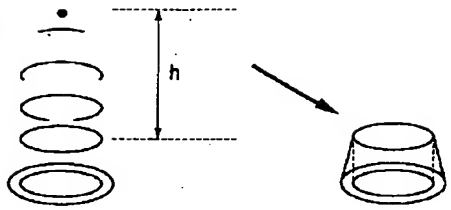

【図１７】



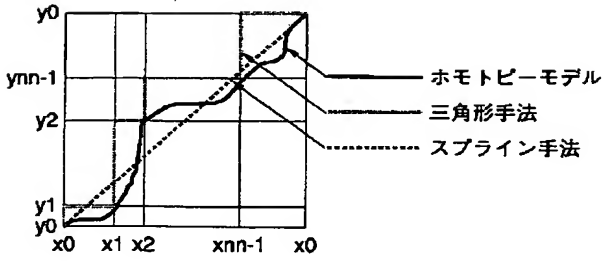
【図２２】



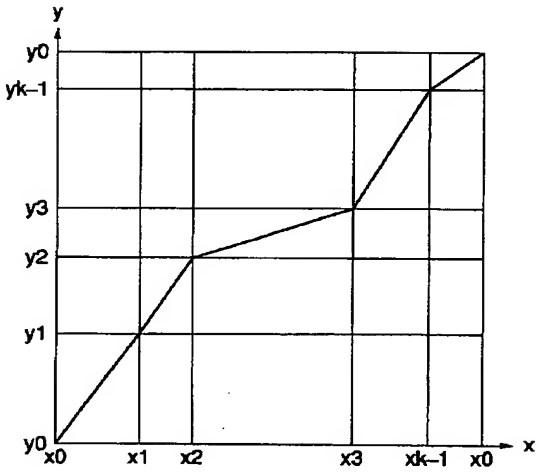

【図２３】



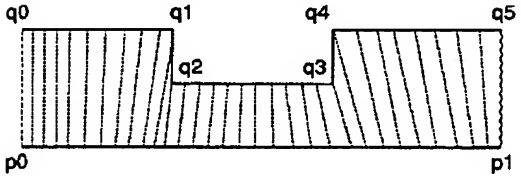

【図２６】



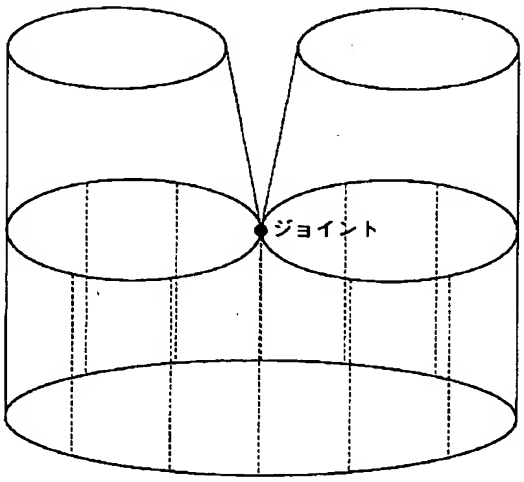

【図２７】



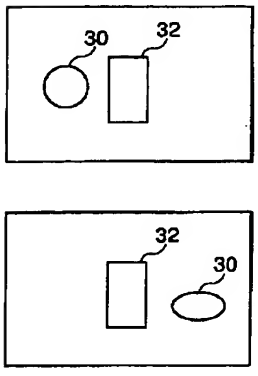

【図２８】



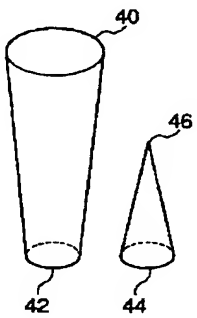

【図２９】



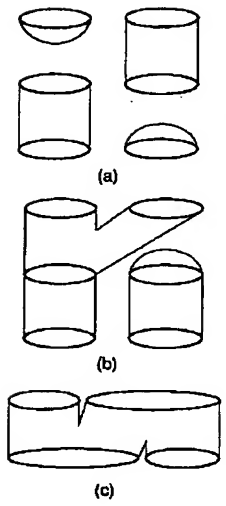

【図３５】



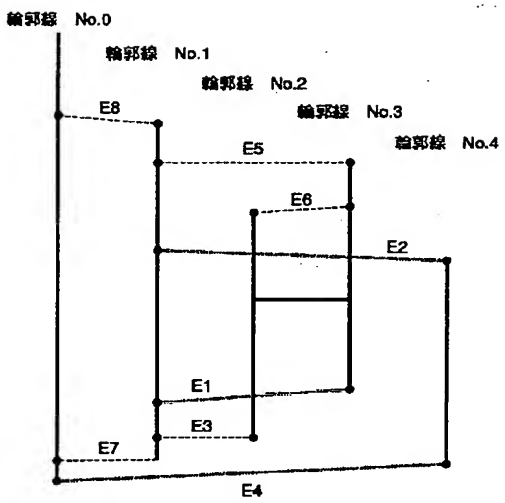

【図４３】



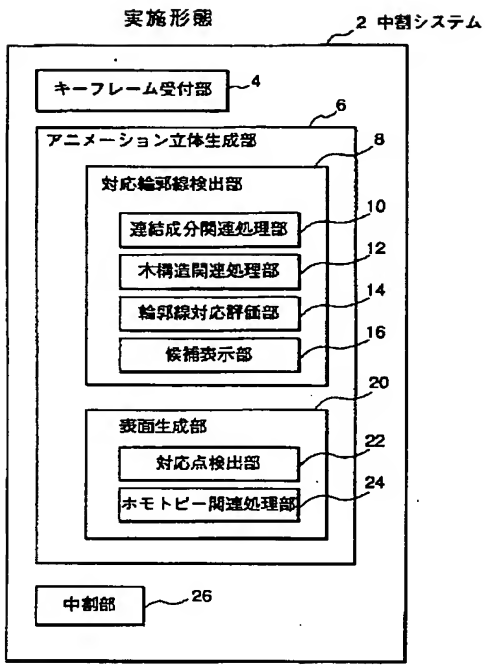
【図３０】



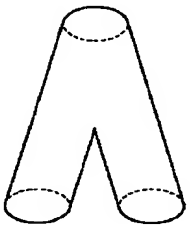

【図３１】



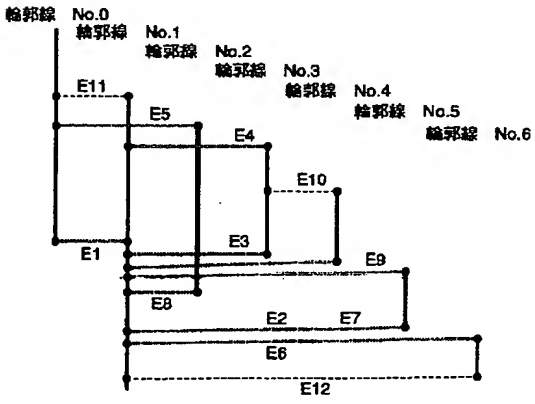

【図３３】



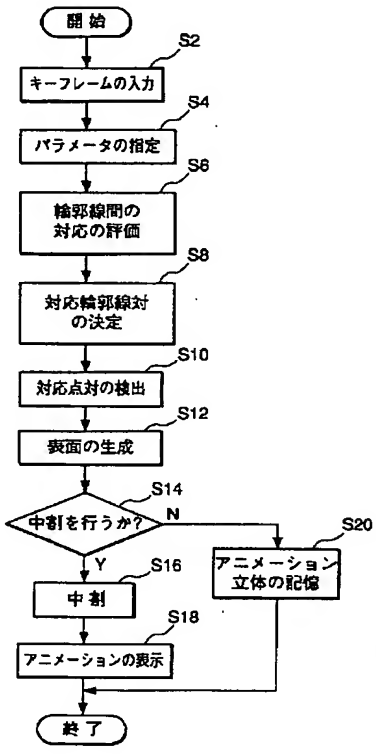

【図６０】



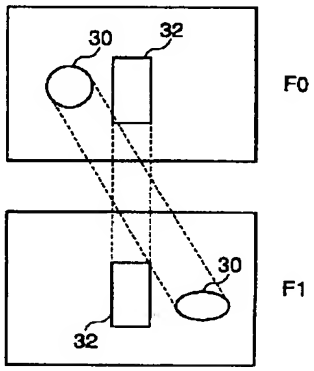
【図３２】



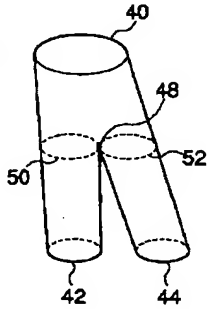

【図３４】



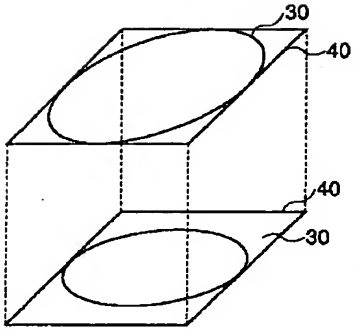

【図３６】



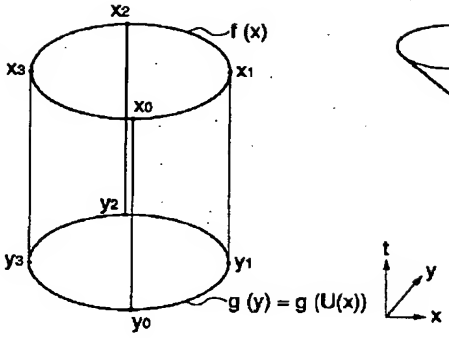

【図４５】



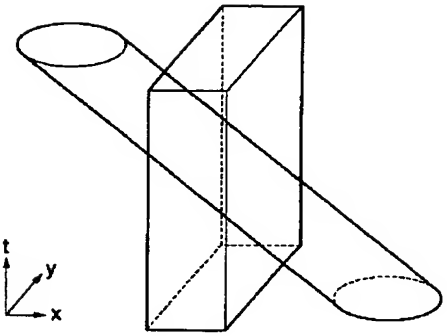
【図３７】



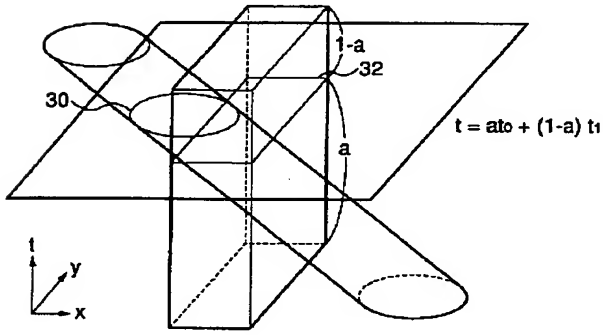

【図３８】



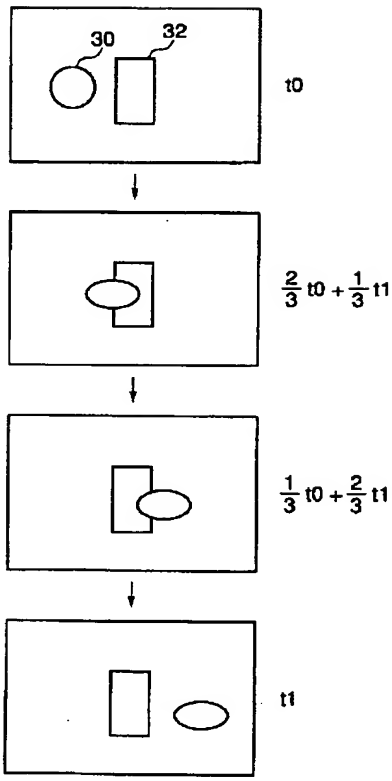

【図３９】



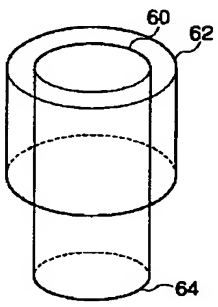
【図４０】



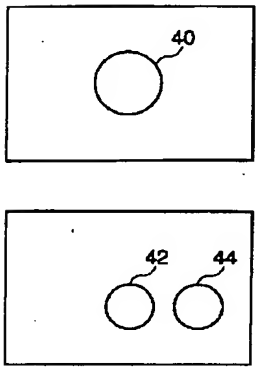

【図４１】



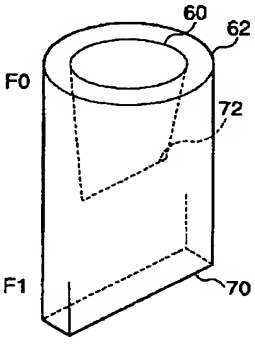

【図４８】



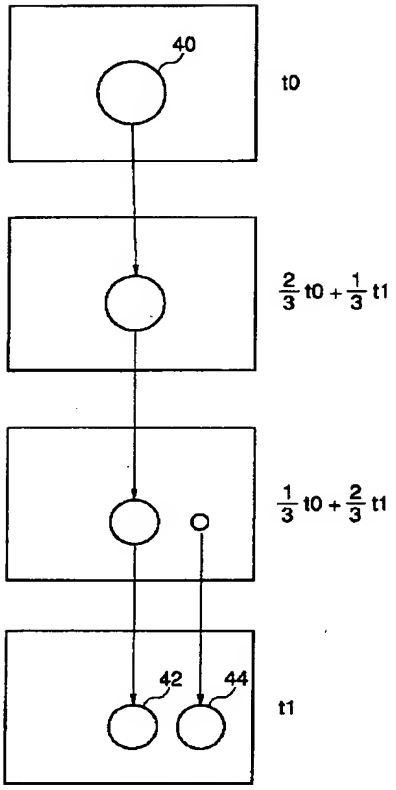

【図４２】



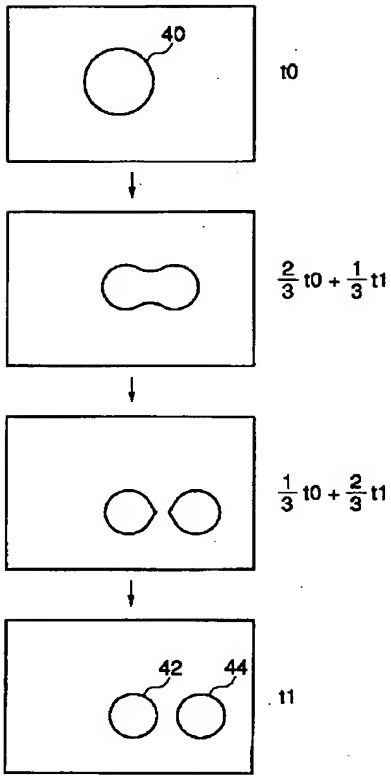

【図５０】



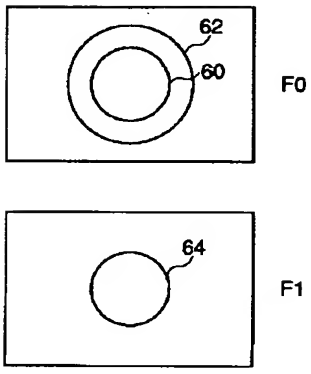
【図４４】



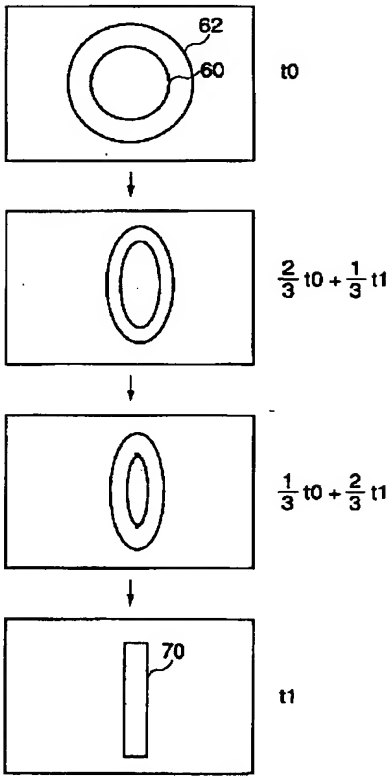

【図４６】



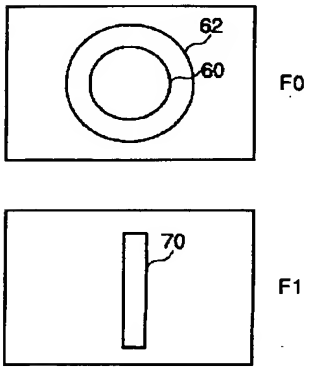

【図４７】



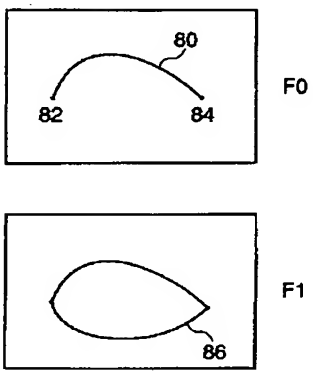

【図５１】



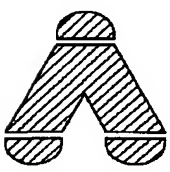

【図４９】



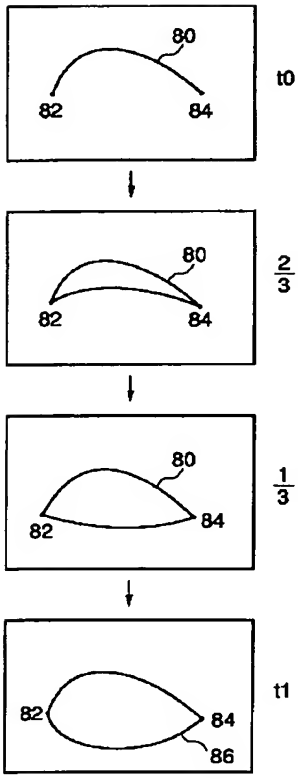

【図５２】



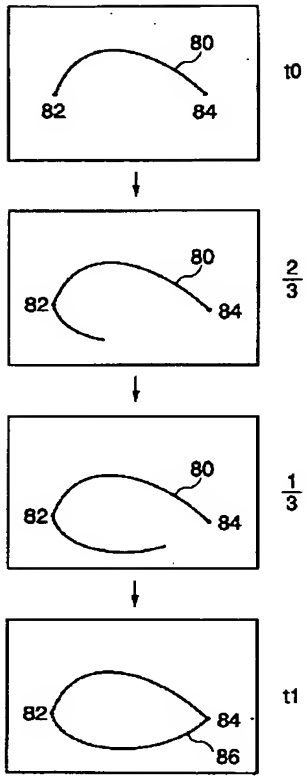

【図６１】



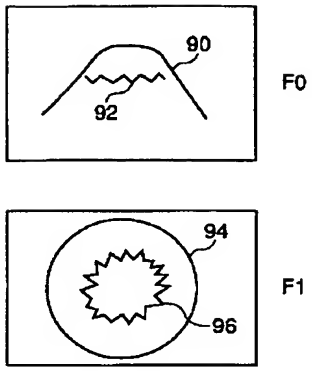
【図５３】



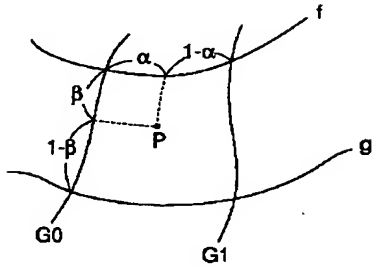

【図５４】



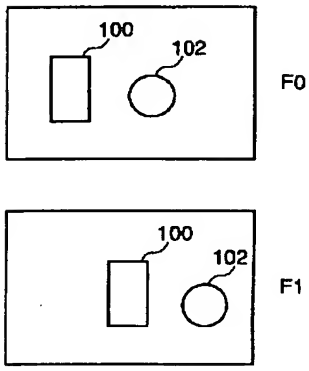

【図５５】



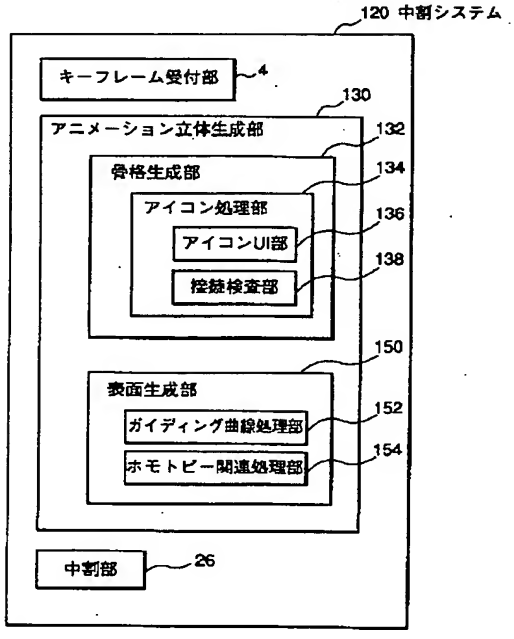

【図５８】



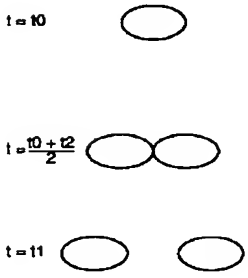

【図５６】



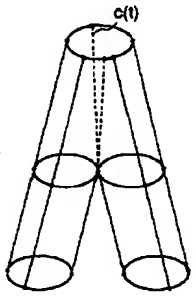

【図５７】



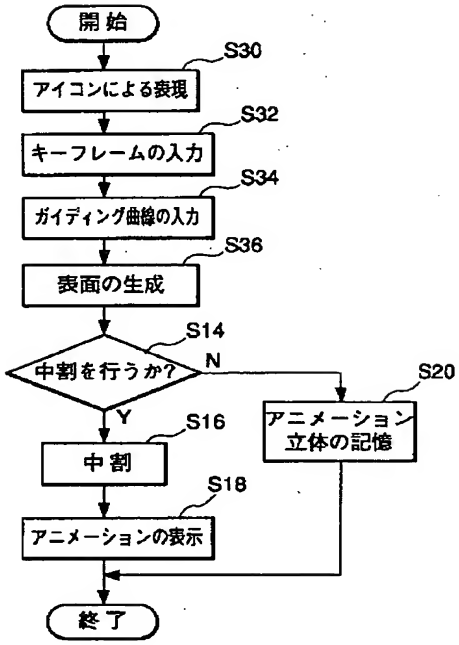

【図６２】




【図６３】



【図５９】


開始
S30
アイコンによる表現
S32
キーフレームの入力
S34
ガイディング曲線の入力
S36
表面の生成
S14
中割を行うか?
N
S20
アニメーション立体の記憶
Y
S16
中割
S18
アニメーションの表示
終了